On the Security of Opportunistic Re-Keying

Stefan Lucks[1], David Schatz[2] and Guenter Schaefer[2]
[1]*Bauhaus-Universität Weimar, Germany*
[2]*Technische Universität Ilmenau, Germany*

Keywords: Opportunistic re-keying, KDF chains, Quantum-resistance

Abstract: Asymmetric cryptography is a cornerstone for security in modern IT infrastructures like virtual private networks (VPNs). Unfortunately, the security of currently deployed schemes is threatened by the ongoing research in quantum computing. And while quantum-resistant alternatives exist, known as post-quantum cryptography (PQC), analyses regarding their (implementation) security are not as mature, yet. Consequently, solely relying on PQC might be susceptible to "store now, decrypt later" attacks. Instead, many researchers suggest using "hybrid" key exchanges, e.g., combining classical asymmetric cryptography, PQC, and symmetric alternatives like quantum key distribution (QKD) and multipath key reinforcement (MKR). In this article, we formalize the idea of "opportunistic re-keying", where a session key is continuously updated using input key material that might be known or even chosen by an attacker. Assuming that at least one input key material is not known to the attacker, we prove the security of the construction in the random oracle model. I.e., when an ideal random function is used for combining the current internal state and new input to generate the next session key and state. Further, we suggest two concrete parameter sets for the construction, corresponding to the security categories 3 and 5 of the NIST standardization process for PQC.

1 INTRODUCTION

Upon existence of sufficiently large quantum computers, "classical" asymmetric cryptography like the RSA cryptosystem and the Diffie-Hellman key exchange will become insecure (Shor, 1997; Proos and Zalka, 2003). And while such quantum computers do not exist yet, attackers can store encrypted traffic now, in the hope to break the underlying Diffie-Hellman exchange at a later point in time ("store now, decrypt later"). Consequently, a transition to *quantum-resistant* alternatives for the exchange of *(symmetric) key material* is required as soon as possible. Note that while quantum computers may also achieve a significant advantage for brute-force attacks on symmetric cryptography by using Grover's algorithm (Grover, 1996) (halving the effective key size), it was shown that brute-force attacks cannot be more efficient (Bennett et al., 1997). Hence, symmetric cryptography will remain secure as long as keys are large enough (≥ 256 bit) and the used algorithms do not show any inherent weaknesses.

One apparently straightforward solution to realize a quantum-resistant key exchange is a transition to post-quantum cryptography (PQC) as a drop-in replacement. The security of PQC *key encapsulation mechanisms (KEMs)* relies on mathematical assumptions which are supposed to hold even with respect to quantum computers. However, the underlying problems, especially for the more efficient candidates (e.g., the ones based on structured lattices), are not studied as well as their "classical" counterparts yet. Similar, the thorough analysis of KEM implementations (e.g., regarding side channel attacks) is still an ongoing process. For this reason, and to implement *cryptographic agility* in general, many security agencies around the world suggest to only use PQC *in addition* to classical asymmetric cryptography for the time being ("hybrid" approaches, e.g., suggested in (Ehlen et al., 2022)).

Schatz et al. recently proposed to go one step further and also integrate orthogonal approaches for the exchange of key material whenever applicable (Schatz et al., 2023; Schatz et al., 2024), which has the advantage of being quantum-resistant even if currently deployed KEMs turn out to be insecure. Such orthogonal approaches include pre-shared keys (PSKs), quantum key distribution (QKD), and multipath key reinforcement (MKR). Furthermore, they argue to maximize the achievable security by *combining* all key materials ever available into one active *session key* instead of only relying on subsets of key materials at a time. To avoid storing all key materials ever collected, they suggest to iteratively combine

Preprint; accepted and in press at SECRYPT 2025, https://secrypt.scitevents.org/

them by using a *KDF chain*. I.e., the current session key and new input key material is combined by using a key derivation function (KDF). A simple synchronization protocol can further detect any mismatch of key material at the two involved parties so that the KDF chains stay in sync. At first sight, such a construction seems to guarantee that the current session key is hidden from attackers as long as they do not know *all* the inputs. However, this intuition was not formally proven yet.

In this article, we formally model the described construction and prove its security in the *random oracle model (ROM)*. I.e., under the assumption that the used KDF behaves like a random oracle, the probability that an attacker is able to disclose the session key is negligible, even if he is able to freely *control* all but one input to the chain. As the underlying function strictly speaking does not have to be a KDF, we prefer the term *opportunistic re-keying* instead of KDF chain in our article. This term is inspired by the core idea to opportunistically use any available key material to refresh (re-key) the current session key, even if the security of the incorporated key material can not be assessed.

The remaining article is structured as follows: In section 2, we recapitulate various approaches for the exchange of key material and argue that in practice, none of them can guarantee to *always* distribute key material in a way that cannot be compromised by attackers in the long run. We proceed by introducing a model of opportunistic re-keying as well as an informal threat model and corresponding security objectives in section 3. Related work regarding the security analysis of similar constructions is discussed in section 4. Subsequently, we define the underlying security game in section 5 and perform our security analysis in section 6. In section 7, we propose a concrete option for the underlying function together with two parameter sets, before we conclude our article in section 8.

2 PRELIMINARIES: EXCHANGE OF KEY MATERIAL

This section recapitulates various quantum-resistant options for exchanging key material between two communication partners. Especially, we focus on possible attacks that leak full or partial knowledge of the exchanged key material, or even can manipulate the key material to the attackers favor.

2.1 Asymmetric Cryptography: Key Encapsulation Mechanisms

A key exchange based on a (PQC) KEM starts with the probabilistic generation of a key pair $(\mathtt{pk}, \mathtt{sk})$ by one of the communication partners. We call this entity the *initiator*. The *public key* $\mathtt{pk}$ is sent to the remote partner (the *responder*), whereas the *secret key* $\mathtt{sk}$ is kept by the initiator. Using the public key as input, the responder uses a probabilistic *encapsulation* algorithm which outputs a shared secret s and a ciphertext c (think: the shared secret in an encapsulated form). The ciphertext is sent back to the initiator, who is then (usually) able to *decapsulate* the same shared secret s based on the knowledge of his secret key $\mathtt{sk}$. Note that some KEMs inherently have a (very small) chance for a decapsulation failure (European Union Agency for Cybersecurity, 2021), which would have to be dealt with accordingly. But for brevity, we will ignore this possibility for the rest of the article.

General Security Discussion for KEMs. Regarding the security of KEMs, the following aspects have to be considered: First, just like the Diffie-Hellman key exchange, KEMs are susceptible to man-in-the-middle attacks if no further *entity authentication* of initiator and responder is performed. One natural solution to achieve entity authentication in this context is to use PQC signature schemes. Second, the security of the currently most promising KEMs are based on hard problems in structured or unstructured lattices, or in coding theory (e.g., see (Ehlen et al., 2022; European Union Agency for Cybersecurity, 2021) for an overview). In contrast to the Diffie-Hellman key exchange, where the security is based on the discrete logarithm problem, it is currently assumed that these problems are intractable by both classical and quantum computers. However, the field of security analysis of KEMs and especially their implementations is not at the same level of maturity as security analysis of the Diffie-Hellman key exchange (Ehlen et al., 2022). Consequently, it might be possible that (quantum or even classical) attackers are able to reconstruct s now or at some later point in time after observing and storing the KEM exchange. This uncertainty especially prevails for the more efficient candidates, namely the ones based on structured lattices.

Entropy vs. Pseudo-Entropy. The functional requirement for a KEM is that at the end, the legit parties agree on a joint secret s of, say, h bit. The security goal of a KEM is that it is infeasible for a computationally bounded adversary, given all data sent between the legit parties, to distinguish s from

a h-bit value u chosen uniformly at random. In the information-theoretical sense, the entropy of s is zero, while the min-entropy of u would be h bit. One possible approach to formalize the security provided to the legit parties by agreeing on the secret s would be the HILL pseudoentropy (Håstad et al., 1999). The HILL pseudoentropy is a complex and powerful formalism to apply an information-theory-like approach to computational security. Arguably, though, in our case there is a simpler approach: We do not need a formalism which would allow us to describe intermediate security levels between entirely secure and completely broken. Either, the KEM at hand fully meets its security goal, or the KEM just fails. If it fails, it might still provide some nontrivial level of security. E.g., the adversary might be able to distinguish s from u, but not to recover s from the data available. But in this case, we just assume the secret s to be compromised (known to or even chosen by the adversary). On the other hand, if the KEM fully meets its security goal, we can replace s by u, while the (computationally bounded) adversary will fail to notice this replacement. I.e., if the KEM is secure, we can just assume a joint random value u with h bits of min-entropy.

2.2 Quantum Key Distribution

With QKD, key material is exchanged by transmitting qubits instead of classical bits, e.g., in the form of (single) photons. The security of QKD relies on the laws of quantum physics, especially the no-cloning theorem (Wootters and Zurek, 1982). I.e., eavesdropping a qubit in transition always introduces a significant chance that the genuine receiver can detect the eavesdropping. However, while there are QKD protocols that are provably secure, e.g., the BB84 protocol (Bennett and Brassard, 2014), security analyses of QKD implementations are not mature yet (Ehlen et al., 2022). Consequently, attackers might be able to infer key material exchanged via QKD, e.g., by utilizing side-channel attacks. Furthermore, attackers might even be able to manipulate QKD keys at will. That is because in the current state of development, the entropy source is implemented in co-located devices, and not on the actual communication devices. E.g., attackers might be able to compromise two QKD devices and pass the same "fake" QKD key material to co-located communication devices.

Note that there are further caveats for the practical deployment of QKD, such as the requirement for an additional, authenticated classical channel, a limited physical reach, and high costs for deployment and operation. However, these are not in the focus here, and we therefore assume for the study of opportunistic re-keying presented in this article that QKD might already be able to achieve a plausible security gain between some communication partners in a network.

2.3 Out-of-band Key Distribution

Instead of exchanging key material *in-band* via the same (or in the case of QKD co-located) channels as the actual communication, one can distribute key material between communication endpoints *out-of-band*. Some options are:

- *Group keys*: While easy to manage, using the same group key for all communication relationships does not provide true end-to-end security: A group key is compromised as soon as at least one group member is compromised.
- *Trusted third party (TTP)*: Key material could be distributed via a TTP, where the communication between the actual endpoints and the TTP is secured by individual PSKs. However, a TTP represents a single point of failure regarding both availability and security of fresh key material.
- *Probabilistic key distribution*: Here (e.g., in (Eschenauer and Gligor, 2002)), the idea is to equip each endpoint with a random subset of a larger key material pool out-of-band, and use in-band communication only to determine mutually known key material. However, this leads to key material being known to more than two endpoints at a time, which is not acceptable in many scenarios.
- *Human key carriers*: A very secure, but also very cumbersome way is to distribute pairwise PSKs using human key carriers. Yet, this does not scale to larger networks with many communication relationships, especially if key material shall also regularly be refreshed.
- *"Automatically deployed" key carriers*: In the context of virtual private networks (VPNs), one could use mobile VPN clients as "automatically deployed" key carriers between VPN gateways, e.g., when moving during business trips (Schatz et al., 2023; Schatz et al., 2024).

All in all, when using out-of-band key distribution, one can not rule out that some PSKs might get known to attackers, e.g., because they managed to compromise the (human or physical) key carrier.

2.4 Multipath Key Reinforcement

Another option for a key exchange that does not rely on asymmetric cryptography is MKR, which has originally been proposed in the context of wireless sensor

networks (Deng and Han, 2008), and more recently in the context of QKD networks (Rass and König, 2011) and VPNs (Schatz et al., 2023; Schatz et al., 2024). With MKR, different key material is sent via multiple paths (e.g., inside a VPN overlay topology) and combined at the endpoints. As long as attackers do not manage to compromise all these paths and a suitable combination function is used, the resulting key may be expected to be unknown to attackers. In the context of this article, one can argue that it suffices to restrict each individual MKR exchange to a single (random) path, and instead shift the combination of key materials from different paths to the construction for opportunistic re-keying. However, we have to expect that some key materials will get known to attackers, e.g., because they managed to compromise an intermediate VPN node on the path.

2.5 Lessons Learned

In summary, there are many promising approaches to exchange key material in a quantum-resistant manner. Further, the different approaches exhibit quite orthogonal attack vectors to compromise individual key materials. Consequently, *combining* multiple approaches via opportunistic re-keying is a natural way to drastically increase the costs of a successful attack. However, to be able to support any kind of key exchange method, our review also revealed that opportunistic re-keying must be robust with regard to input key material that is known or even manipulated by attackers.

3 THE MODEL OF OPPORTUNISTIC RE-KEYING

Opportunistic re-keying is the approach to gather several input keys via different channels and employing different techniques, none of which being fully trusted, and apply some transformation to generate one secret session key, which can be used as a "master key" for subsequent communication. We will implement this as a continuous process: Each time a new input key arrives, the internal state is updated and a fresh session key is generated. Whenever parties need to communicate securely, they will use the most recent session key.

3.1 Notation

By $\Pr[A]$, we denote the probability of an event A to occur, and $\overline{A}$ is the event of A to not occur, thus $\Pr[A]+\Pr[\overline{A}]=1$. If $\Pr[A]>0$ we define the conditional probability of event B if A occurs by $\Pr[B|A]$.

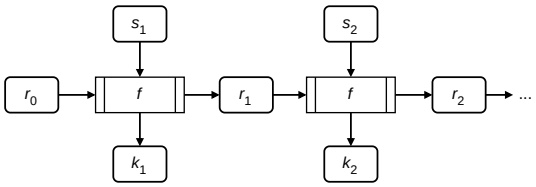

Figure 1: Opportunistic re-keying dynamically combines input key materials $s_j, 1 \le j \le i$ to one session key k_i. The proposed scheme realizes this by iteratively combining an internal state r_j with the next available key material s_{j+1} to derive the next state r_{j+1} and the next session key k_{j+1}, calling a function f.

If S is a finite set, we write $s \leftarrow_{\$} S$ for choosing s uniformly at random from S. If D_S is a distribution over a set S, $\Pr_{D_S}[s]$ is the probability to sample s under D_S. If for all $s \in S$ it holds that $\Pr_{D_s}[s] \le 2^{-h}$, then the *min-entropy* of D_S is at least h. For $s \in S$ we write $s \leftarrow^{h} S$ to indicate s being sampled to some distribution with min-entropy h.

As inspired by the abstract construction in (Schatz et al., 2023) and (Schatz et al., 2024) and depicted in Figure 1, opportunistic re-keying may be summarized as follows: Two communication partners (Alice and Bob) receive an identical sequence of input key materials $s_1, \ldots, s_i$. As discussed in section 2, each individual key material $s_j, 1 \le j \le i$ is either determined by Alice, Bob, or an external key source. For this, we assume that each entropy source that is *not controlled by attackers* shows at least h bit of min-entropy. Furthermore, Alice and Bob share a joint initial state r_0, e.g., a group key. Starting from this, opportunistic re-keying produces a sequence of session keys k_i and states r_i both as a function of the initial state and all respective input key materials $s_j, 1 \le j \le i$.

To allow for an efficient dynamic generation of new session keys without having to store all individual s_j, this function is realized iteratively as follows:

$$(r_{j+1}, k_{j+1}) \leftarrow f(r_j, s_{j+1}) \qquad (1)$$

I.e., a function f is used to combine the current state r_j and the next input key material s_{j+1} to generate the next state r_{j+1} and the next session key k_{j+1}. For the remaining article, we assume κ bit session keys and internal states of size ρ bit. We do not fix the size of the input key material, i.e., formally $s_i \in \{0,1\}^*$. Still, we require at least h bit of min-entropy for input keys below, which implies $|s_i| \ge h$. Thus, the function f has the following signature:

$$f : \{0,1\}^{\rho} \times \{0,1\}^* \to \{0,1\}^{\rho} \times \{0,1\}^{\kappa} \qquad (2)$$

Note that the original proposal in (Schatz et al., 2023) and (Schatz et al., 2024) does not distinguish between an internal state and the session key. Instead, they suggest to persistently store a derivation

of the session key ("recovery key") to reduce the risk of leaking old session keys. The construction proposed and studied in this article, in contrast, enables a straightforward way to persist the current state, which is required when either Alice or Bob had to reboot their device and want to "recover" the level of security previously achieved via their pairwise key and state sequence.

3.2 Informal Introduction of Threat Model And Security Properties

To provide an intuition for the reader, we first informally introduce the threat model of opportunistic re-keying. A formal model will be provided in section 5. For ease of reading, we prefer to personalize the adversary and refer to her as "Eve". We care about two security notions: resilience and forward security.

Resilience. We allow Eve to taint input keys and to compromise states:

- Eve can either learn the input key s_j ($\rightarrow$ sections 2.3 and 2.4) or even choose s_j ($\rightarrow$ sections 2.1 and 2.2). We assume her to choose s_j, as this represents the stronger attack capability. We refer to this as *tainting*.

  Untainted input keys s_i are randomly generated strings with h bits of min-entropy.[1]
- Furthermore, for any j of her choice, Eve can read the state r_j. We refer to this as a "compromise". As special case, we assume the initial state r_0 to always be compromised. Otherwise, Alice and Bob would start the communication with a joint secret key unknown to the adversary, which would, kind of, eliminate the need for opportunistic re-keying.

  Note that Eve "knows" any compromised state, but a state may also be "known" to her even if she did not compromise that state. Namely, if the state r_{i-1} is known (compromised or otherwise) and if the matching input key s_i is tainted, then the state r_i derived from r_{i-1} and s_i is "known", as Eve can compute it on her own.

The purpose of opportunistic re-keying (or, of re-keying in general) is to generate secure keys, even when many (but not all) of the external inputs are tainted and many (but not all) of the internal states are compromised:

- We refer to a session key k_j as secure, if it is infeasible for Eve to distinguish k_j from a uniform random string of the same size. In that case, it is obviously safe for Alice and Bob to employ k_j for symmetric encryption and authentication.
- For every $\alpha \geq 1$ and $\beta \geq \alpha$, we require the session key k_β to be secure if
  - Eve did neither taint s_α,
  - nor did she read any of the states $r_\alpha, \ldots, r_{\beta-1}$.

An opportunistic re-keying scheme that fulfills this property is *resilient*.

Forward security. Another common requirement for key generation schemes is *forward security* (also known as backtracking resistance, e.g., in (Barker and Kelsey, 2015)): If a session key k_j is considered secure at some point of time, then subsequent compromises (or tainting events) of data generated jointly with k_j, or generated or used later, must not enable the adversary to break the security of k_j. So the following data might be compromised or tainted, without endangering the security of k_j:

- r_j, which is generated jointly with k_j, and
- $r_{j+1}, s_{j+1}, r_{j+2}, s_{j+2}, \ldots$, which are all used/generated after generating k_j.

Resilience implies forward security. Assume the scheme is not forward secure. I.e., the security of k_j can be broken by compromising or tainting one or more of the values r_j, r_{j+1}, s_{j+1}, r_{j+2}, s_{j+2}, ..., but none of the other values. As those "other values" are $s_j, r_{j-1}, s_{j-1}, r_{j-2}, s_{j-2}, \ldots$, this very scheme is not resilient, either. Below, we will thus formalize and prove resilience. We do not need any separate argument to prove forward security.

3.3 Random Oracles

For complex cryptographic systems, it is common to analyze the security of a cryptosystem in the random oracle model, i.e., to assume an underlying function to behave like an ideal random function. Below, we will make this assumption. Namely, we will treat the function

$$f : \{0,1\}^\rho \times \{0,1\}^* \rightarrow \{0,1\}^\rho \times \{0,1\}^\kappa$$

as a random oracle. The idea, depicted in Algorithm 1, is to initially assume an undefined f and to keep track of pairs $(x, f(x))$. If, for a given x, no such pair is in the list, $f(x)$ is chosen uniformly at random.

[1] Usually, secret keys transmitted by some key exchange mechanism are required to be uniformly distributed. As opportunistic re-keying has to deal with many different and unspecified techniques to transmit secret input keys, we relax this requirement.

Algorithm 1: Defining f as a random oracle. $\mathcal{T}$ holds a map from $\{0,1\}^\rho \times \{0,1\}^*$ to $\{0,1\}^\rho \times \{0,1\}^\kappa$. Initially, $\mathcal{T}$ is empty.

```
T ← {};
function f(x,y)
    if (x,y) ∉ T then
        T[x,y] ←$ {0,1}^(ρ+κ)
    return T[x,y]
```

4 RE-KEYING: RELATED WORK

Our approach is based on using a symmetric primitive to derive a sequence of secret internal keys and/or states. Each key/state is only used once. Re-keying is a common approach in symmetric cryptography, for a lot of different purposes. Some examples are the following:

- PBKDEF2 (Password-Based Key-DErivation Function 2) takes a password as the initial secret key, and iterates a symmetric primitive to eventually generate a pseudorandom output (Turan et al., 2010). The purpose of re-keying is to slow down attacks which attempt to exploit a low entropy of the initial password.
- Like opportunistic re-keying, the double ratchet algorithm is about providing pseudorandom session keys (Perrin and Marlinspike, 2016). It combines two "ratchets", one based on symmetric cryptography, and a second one on asymmetric cryptography. Firstly, double-ratcheting provides forward security: in the case of a compromise, previous keys are safe. Secondly, it ensures plausible deniability: If keys are only used once, and then immediately replaced by a new key, authors of encrypted or authenticated documents can easily deny their authorship.
- Forward security has also been the purpose of re-keying in the Bellare-Yee PRG (pseudorandom generator) (Bellare and Yee, 2003): Let $F : \{0,1\}^n \to \{0,1\}^n \times \{0,1\}^n$ be a PRG. To generate a stream of ℓ pseudorandom n-bit values $X_1, X_2, \ldots X_\ell$, one starts with an initial secret $K_0 \in \{0,1\}^n$ and then iterates the operation $(K_i, X_i) \leftarrow F(K_{i-1})$, for $i \in \{1, \ldots, \ell\}$. If, say, K_j has been compromised, then $K_0, \ldots, K_{j-1}$ and, by extension, the pseudorandom values $X_1, \ldots, X_{j-1}$ remain safe.
- Other authors proposed the Bellare-Yee generator for its leakage-resilience (Standaert et al., 2010). Assume that some information about K_j leaks (1) when $F(K_{j-1})$ is called to compute K_j and (2) when $F(K_j)$ is called. If this amount of leakage does not suffice to actually find K_j, the adversary will not find K_j at all, because the generator advances to K_{j+1}. The proof from (Standaert et al., 2010) is in some variant of the random oracle model. In the standard model the Bellare-Yee generator is insecure against "future computation attacks" (Dziembowski and Pietrzak, 2008; Pietrzak, 2009). Such attacks are considered impossible in any practical setting. But, it seems to be difficult to make any standard-model assumptions on F to prove resistance against future computation attacks.
- The Pietrzak-PRG has been proven leakage-resilient in the standard model (Pietrzak, 2009), though with poor concrete security bounds. It employs a function $F : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^k \times \{0,1\}^n$ as the underlying primitive. Given a random initial value $X_0 \in \{0,1\}^n$ and a secret key pair $(K_0, K_1) \in \{0,1\}^k \times \{0,1\}^k$, it computes a sequence of pseudorandom outputs $X_0, X_1, \ldots, X_{\ell+1}$ by iteratively calling $(K_{i+1}, X_i) \leftarrow F(K_{i-1}, X_{i-1})$, for $i = 1, 2, \ldots$
- Finally, the XDRBG significantly resembles our construction (Kelsey et al., 2024). Like our construction, the XDRBG derives its new internal state from the current internal state and from some additional input. In the case of the XDRBG, the additional input is a random seed from some entropy source, in our case, it is a key sent via some key exchange mechanism. Both the XDRBG and our construction require the additional input to provide a certain amount of min-entropy. The purpose of the XDRBG is to provide pseudorandom session keys from random sources, typically from physical random sources such as a Zener Diode or a Geiger counter, the purpose of opportunistic re-keying is to provide pseudorandom session keys from the output of some out-of-band distributed keys of unknown trustworthiness.

5 THE OPPORTUNISTIC RE-KEYING GAME

In subsection 3.2, we informally introduced the notion of resilience in the presence of an adversary Eve, who is allowed to taint ingoing data and to compromise internal states. To properly formalize this setting, we define the opportunistic re-keying game, see Algorithm 2. This algorithm assumes a total of R steps to be performed. Essentially, R is the "data complexity",

Algorithm 2: Opportunistic Re-Keying Game:
The adversary, whom we personalize as "Eve", is interacting with a challenger. The initial state is $r_0 = X_0$ and compromised. The game runs for R steps. In each step, Eve can either taint the input key, or make an untainted step. Regardless of tainting the input key or not, she can also compromise the outgoing state. Each of the input keys $s_1, s_2, \ldots, s_R$, if not tainted, is distributed according to a distribution with min-entropy at least h. To keep track of compromised and known states, arrays $C[0], \ldots, C[R]$ and $K[0], \ldots, K[R]$ are maintained. $C[i] = 1$ indicates compromised, and $K[i] = 1$ indicates known. The array $T[1], \ldots, T[R]$ is used similarly: $T[i] = 1$ indicates tainted. At the end, Eve tries to guess the secret bit b.

```
1  function Challenger(X_0, R, h)
       /* Start the game: Randomly choose secret
          bit b and set the initial state                */
2      b ←$ {0,1};
3      r_0 ← X_0;
       /* r_0 is compromised and known                    */
4      send r_0 to Eve; C[0] ← 1; K[0] ← 1;
       /* Perform R steps of the game                     */
5      for step i ∈ {1,...,R} do
6          if Eve taints s_i by Y_i then
7              s_i ← Y_i; T[i] ← 1;          /* Tainted */
8          else   /* Eve makes an untainted step */
9              s_i ←^h {0,1}*; T[i] ← 0;
10         (r_i, k_i) ← f(r_{i-1}, s_i);
11         if Eve compromises state r_i then
               /* Compromised and known                   */
12             send r_i to Eve; C[i] ← 1; K[i] ← 1;
13         else
14             C[i] ← 0;   /* Uncompromised ...          */
15             if K[i-1] = 1 and T[i] = 1 then
16                 K[i] ← 1;        /* ... but known     */
17             else
18                 K[i] ← 0;  /* ... and unknown         */
19         if (K[i-1] = 0 or T[i] = 0) and b = 0
           then
20             Z ←$ {0,1}^κ;
21             send Z to Eve;
22         else
23             send k_i to Eve;
       /* Finish the game                                 */
24     Receive b* from Eve;
25     Eve wins ⟺ b* = b;
```

i.e., the adversary will not observe more than R session keys. The algorithm represents Eve's capabilities to taint and compromise:

- Eve can choose to taint an input key (cf. line 7). The algorithm keeps track of which input key has been tainted via the array $T[1], \ldots, T[R]$; here $T[i] = 1$ indicates tainted and $T[i] = 0$ indicates untainted.
- Eve can compromise states (cf. line 12); this time, the array $C[0], \ldots, C[R]$ is used to keep track of which state has been compromised. $C[i] = 1$ indicates a compromised r_i, $C[i] = 0$ an uncompromised r_i.

Every compromised state is known to the adversary, but even if r_i is not compromised, it may be known to the adversary. Namely, if r_{i-1} is known and s_i is tainted, Eve can compute r_i on her own. Thus, we assume r_i to be known to the adversary, indicated by $K[i] = 1$. For this purpose, the algorithm employs the array $K[1], \ldots, K[R]$, cf. lines 4, 12, 16, and 18.

If Eve can compute r_i on her own, she can also compute the matching session key k_i. An opportunistic re-keying scheme is resilient, if Eve cannot distinguish the remaining outputs – i.e., the outputs she cannot compute on her own – from uniformly distributed random values. The algorithm matches this as follows:

1. Initially, in line 2, the challenger randomly chooses a secret bit b.
2. During the steps of the game, whenever Eve cannot compute the session key k_i on her own[2], the challenger replaces k_i by a random value – but only if $b = 0$, as done in line 20. If Eve can compute the session key on her own, or if $b = 1$, the challenger sends the proper session key to Eve, cf. line 23.
3. Finally, in line 24, Eve attempts to guess b.

It is trivial for Eve to succeed with probability $\frac{1}{2}$: Flip a fair coin and output a random bit as the guess for b. An opportunistic re-keying scheme is resilient, if Eve's probability does not exceed the probability $\frac{1}{2} + \varepsilon$ for some small ε.

Eve is allowed to query the random oracle f on her own. Assume Eve to make up to Q oracle-queries. In the random oracle model, one usually considers this value Q as the "query complexity", indicating a lower bound for the actual run-time of an attack. Below, we will provide an upper bound for ε, which depends on Q and on the data complexity R – and on the parameters h and ρ.

[2]This is the case when r_{i-1} is unknown to her, or when she did not taint s_i, cf. line 19.

6 SECURITY ANALYSIS OF OPPORTUNISTIC RE-KEYING

Theorem 1. *Let Eve be an adversary who runs* ***function*** CHALLENGER(X_0, R, h) *from Algorithm 2. She iterates the main loop of the algorithm for R times. The min-entropy for each non-tainted input key s_i is at least h bit, $h \leq \rho$. If Eve makes up to Q queries to the random oracle on her own, her probability to win the game in Algorithm 2 is* $\Pr[\textit{Eve wins}] \leq \frac{1}{2} + \varepsilon$ *with*

$$\varepsilon \leq \frac{R^2}{2^{\rho+1}} + \frac{Q}{2^h} + \frac{QR}{2^\rho}. \tag{3}$$

Proof. For the proof, reconsider line 10 of Algorithm 2, which invokes the random oracle $f(\cdot,\cdot)$ and computes r_i and k_i as follows: $(r_i, k_i) \leftarrow f(r_{i-1}, s_i)$. By the definition of $f(\cdot,\cdot)$ in Algorithm 1, r_i and k_i are uniformly distributed random values, and, if no other call to f with exactly the same input parameters r_{i-1} and s_i is made, r_i and k_i are independent from all other values chosen or computed when running the game. In this case, $\varepsilon = 0$, since Eve cannot distinguish the uniform random value k_i from the uniform random value Z chosen in line 20 of the algorithm. Hence, Eve can only win with any probability exceeding $\frac{1}{2}$, if, for some i, another call for $f(x,y)$ is made during the course of the game, with $x = r_{i-1}$ and $y = s_i$. This means

$$\varepsilon \leq \Pr[\text{BAD1}] + \Pr[\text{BAD2} | \overline{\text{BAD1}}]$$

for the following two "bad events":

BAD1: $\exists i, j,\ 1 \leq i < j \leq R$: $r_i = r_j$;

BAD2: there exists an unknown $x = r_{i-1}$ or an untainted $y = s_i$, such that Eve queries the random oracle for $f(x,y)$.

Informally, BAD1 covers the case that the game might accidentally cycle, and BAD2 covers the case that Eve manages to guess $x = r_{i-1}$ or $y = s_j$ and asks the random oracle for $f(x,y)$. Regarding BAD2, however, guessing x and y and asking for $f(x,y)$ is pointless for Eve *except* when these x and y are used to generate a value k_i which is either sent back to Eve or replaced by a random Z in line 20. This is the case when r_{i-1} is unknown to Eve, or when Eve did not taint s_i, cf. footnote 2 and line 19.

To bound $\Pr[\text{BAD1}]$, observe that the game essentially chooses random r_1, r_2, ..., until eventually BAD1 occurs or until the game terminates. So $\Pr[\text{BAD1}]$ is a "birthday bound":

$$\Pr[\text{BAD1}] \leq \sum_{i=1}^{R} \frac{i}{2^\rho} \leq \frac{R^2}{2^{\rho+1}}. \tag{4}$$

For $\Pr[\text{BAD2} | \overline{\text{BAD1}}]$, we distinguish two cases:

1. $\overline{\text{BAD1}}$ and (s_i is untainted),
2. $\overline{\text{BAD1}}$ and (s_i is tainted) and (r_i is unknown).

In the first case, even if Eve chooses a known r_{i-1}, she has to guess a unique s_i, which is a random value with h bit of min-entropy. From $\overline{\text{BAD1}}$ we know $r_{i-1} \neq r_{j-1}$, for $i \neq j$, thus, querying for $f(r_{j-1}, s_i)$ will not trigger the event BAD2. Thus, for each single query to the f-oracle made by Eve, the event BAD2 occurs with a probability $\leq 2^{-h}$.

In the second case, there are R values s_1, ..., s_R, and, if tainted, they can all be the same. So Eve may not have to guess a unique r_{i-1}: there are at most R unknown states r_{i-1}, such that Eve might query for $f(r_{i-1}, s_i)$. As each unknown r_{i-1} is a random ρ bit value, the probability for this event to occur is $\leq \frac{R}{2^\rho}$.

In total, Eve queries the random oracle Q times, so

$$\Pr[\text{BAD2} | \overline{\text{BAD1}}] \leq Q \cdot \left(\frac{1}{2^h} + \frac{R}{2^\rho} \right). \tag{5}$$

The claimed result in Equation 3 is the sum of the bounds from Equation 4 and Equation 5. □

Post-quantum security. Our result can be interpreted as follows: If the amount of data observed by Eve is not too large, i.e., if $R \ll 2^{\rho/2}$, then the first term of Equation 3 is negligible. If, furthermore, $R \ll 2^{\rho-h}$, the third term of the same equation is also negligible. In this case, to succeed with significant probability, Eve needs to find a preimage with min-entropy at least h bit. A classical adversary needs to make at least $Q \approx 2^h$ queries to the random oracle. A quantum oracle, running Grover's algorithm, needs to make at least about $\sqrt{2^h}$ queries.

Multi-party security. To rephrase our result, assume a pair (A,B) of communicating parties and assume the number $R_{(\text{A,B})}$ of keys generated by A and B to communicate is not too large. Given about $n^2/2$ pairs in an n-party network, it is straightforward to conclude that Eve, if she cares about breaking any connection between any two parties, rather than trying to break into a specific connection, cannot improve her performance by more than a factor $n^2/2$.

Actually, the factor $n^2/2$ is too optimistic, from Eve's point of view. Require an initial choice of disjoint random initial states r_0 for each of the communicating pairs of parties. Recall the proof of Theorem 1 and define the event BAD1 to include cross-pair state collisions. I.e., BAD1 also occurs if the internal state of two different connections happens to be the same. Thus, we (re-)define the data complexity R to be the sum of all steps performed by all pairs of parties. As before, assume R to be not too large, i.e., $R \ll 2^{\rho/2}$

		P1	P2
Data complexity	$\log_2(R)$	32	60
State size	ρ	256	512
Min-entropy	h	192	256
Query complexity	$\log_2(Q)$	192	256
NIST category		3	5

Table 1: Proposed parameter sets for instantiations of the opportunistic re-keying scheme from Figure 1. As elaborated in the text, P1 is the "normal" security level, which matches NIST category 3 security (National Institute of Standards and Technology, 2016) in typical application scenarios (100 re-keying events per second, and observation time 16 months), while P2 is a very conservative choice matching NIST category 5 security, even in a network with one million re-keying events per second and the attack running for about 37 000 years.

and $R \ll 2^{\rho-h}$. In this case, a quantum adversary *Eve still needs to make about* $\sqrt{2^h}$ *queries to break into any of the about* $n^2/2$ *connections*.

Avoiding the random oracle model. For the analysis above, we did model f as a random oracle. This is a common approach in cryptography, but many researchers prefer standard model proofs over proofs in the random oracle model.

So, is it possible to prove the construction from Figure 1 secure in the standard model? In our random oracle proof, we actually assume that, given $(r_j,k_j) = f(r_{j-1},s_j)$ and either r_{j-1} or s_j, it is infeasible to find the remaining input value (i.e., either s_j or r_{j-1}). If we restrict the signature of f to $f : \{0,1\}^\rho \times \{0,1\}^h \to \{0,1\}^\rho \times \{0,1\}^\kappa$ and assume the input keys to be uniform random h-bit values, then this is precisely the requirement for f to be a dual PRF (Bellare and Lysyanskaya, 2015). Thus, it may indeed be possible to prove the security of our construction without relying on the random oracle model. But we believe the disadvantages of such an approach would outweigh the sole advantage of getting rid of the random oracle. The disadvantages would include (1) a more complex and less intuitive security proof, (2) the restriction for the untainted input keys to be uniformly distributed fixed-size values, and (3) the reliance on a new and not yet well-understood security notion, namely on the security of a dual PRF.

7 CONCRETE PROPOSALS

For the sake of concreteness, we propose two sets of parameters, see Table 1. Parameter set P1 should suffice for most applications. P2 is very conservative, and we recommend it for high-security applications.

Comments on parameter set P1. If we assume a network with 100 re-keying events per second, this makes $100*60*60*24*365 \approx 2^{31.55}$ such events per year. To collect $R = 2^{32}$ such events, Eve has to observe the network for about 16 months without missing any of the events. The min-entropy $h = 192$ means that on a classical computer, she needs to perform about 2^{192} calls of the function $f(\cdot,\cdot)$. On a quantum computer, she needs about 2^{96} iterations of Grover's algorithm. Thus, parameter set P1 matches category 3 security from NIST.

Comments on parameter set P2. Maybe we need an even higher security, we assume more re-keying events per second in our network, or we assume Eve to be a lot more patient with observing the network. So assuming a very fast network performing one million re-keying events per second, the number of re-keying events per year is $1\,000\,000*60*60*24*365 \approx 2^{44.84}$. To collect $R = 2^{60}$ re-keying events, Eve needs to observe the network for about $2^{60-44.8} \approx 37\,000$ years. The min-entropy $h = 256$ means that we maintain category 5 security in the NIST classification (attack time about 2^{256} classical operations, or about 2^{128} Grover iterations).

Instantiation based on SHAKE-256. For a concrete instantiation, we propose the SHAKE-256 XOF (eXtended Output Function), which has been derived from SHA-3 (Kelsey et al., 2016). A XOF essentially takes any number of input bits (as much as provided) and generates as many output bits as required.

In our case, we need a function $f : \{0,1\}^\rho \times \{0,1\}^* \to \{0,1\}^\rho \times \{0,1\}^\kappa$ for whatever choice of ρ and κ we propose. For parameter set P1, we suggest $\rho = 256 \geq h = \kappa = 192$, for parameter set P2 $\rho = 512 \geq h = \kappa = 256$. Regarding the performance of SHAKE-256 for parameter set P2, an implementation in Rust[3] requires ≈ 19000 instructions (≈ 850 ns) per invocation on an AMD Ryzen 7 5700G processor.

8 CONCLUSIONS

Opportunistic re-keying is a future-proof way to provide quantum-resistant session keys for symmetric encryption in IT infrastructures, such as VPNs. Compared to independently using (the same or different) key exchange mechanisms for individual session keys, opportunistic re-keying has two key advantages:

[3]Using the `sha3` crate, `v0.10.8`, for the implementation. Instruction count provided by the `iai` crate, `v0.1.1`.

1. It allows an early adoption of new key exchange mechanisms, even if analysis of their (implementation) security is not mature, yet, such as the more efficient PQC candidates and QKD.
2. It allows using mechanisms that have an inherent chance of sometimes generating a key that gets known to attackers, such as keys exchanged via mobile devices during business trips, MKR, and probabilistic key management.

Further, the sheer amount of different deployed key exchange mechanisms and their frequency of execution may drastically increase the overall effort for attackers to compromise the current session key.

ACKNOWLEDGEMENTS

This research is partially funded by dtec.bw – Digitalization and Technology Research Center of the Bundeswehr [project MuQuaNet]. dtec.bw is funded by the European Union - NextGenerationEU.

REFERENCES

Barker, E. B. and Kelsey, J. M. (2015). Recommendation for Random Number Generation Using Deterministic Random Bit Generators. Technical Report NIST SP 800-90Ar1, National Institute of Standards and Technology.

Bellare, M. and Lysyanskaya, A. (2015). Symmetric and dual PRFs from standard assumptions: A generic validation of an HMAC assumption. *Cryptology ePrint Archive*. https://eprint.iacr.org/2015/1198.

Bellare, M. and Yee, B. S. (2003). Forward-security in private-key cryptography. In *Topics in Cryptology - CT-RSA*, volume 2612 of *LNCS*, pages 1–18. Springer.

Bennett, C. H., Bernstein, E., Brassard, G., and Vazirani, U. (1997). Strengths and Weaknesses of Quantum Computing. *SIAM J. Comput.*, 26(5):1510–1523.

Bennett, C. H. and Brassard, G. (2014). Quantum cryptography: Public key distribution and coin tossing. *Theoretical Computer Science*, 560:7–11.

Deng, J. and Han, Y. (2008). Multipath Key Establishment for Wireless Sensor Networks Using Just-Enough Redundancy Transmission. *IEEE TDSC*, 5(3):177–190.

Dziembowski, S. and Pietrzak, K. (2008). Leakage-resilient cryptography. In *49th Annual IEEE FOCS*, pages 293–302. IEEE Computer Society.

Ehlen, S., Hagemeier, H., Hemmert, T., Kousidis, S., Lochter, M., Reinhardt, S., and Wunderer, T. (2022). Quantum-safe cryptography – fundamentals, current developments and recommendations. Technical report, Federal Office for Information Security (BSI). https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Brochure/quantum-safe-cryptography.html?nn=916626.

Eschenauer, L. and Gligor, V. D. (2002). A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM CCS*, pages 41–47.

European Union Agency for Cybersecurity (2021). Post-Quantum Cryptography: Current state and quantum mitigation. Technical report, Publications Office of the European Union.

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th ACM STOC*, pages 212–219.

Håstad, J., Impagliazzo, R., Levin, L. A., and Luby, M. (1999). A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396.

Kelsey, J., Chang, S., and Perlner, R. (2016). SHA-3 derived functions: cSHAKE, KMAC, TupleHash and ParallelHash. Technical report, National Institute of Standards & Technology.

Kelsey, J., Lucks, S., and Müller, S. (2024). XDRBG: A Proposed Deterministic Random Bit Generator Based on Any XOF. *IACR Transactions on Symmetric Cryptology*, 2024(1):5–34.

National Institute of Standards and Technology (2016). Submission requirements and evaluation criteria for the post-quantum cryptography standardization process.

Perrin, T. and Marlinspike, M. (2016). The Double Ratchet Algorithm. Technical Report Revision 1. https://signal.org/docs/specifications/doubleratchet.

Pietrzak, K. (2009). A leakage-resilient mode of operation. In *Proceedings of EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 462–482. Springer.

Proos, J. and Zalka, C. (2003). Shor's discrete logarithm quantum algorithm for elliptic curves. *Quantum Information and Computation*, 3(4):317–344.

Rass, S. and König, S. (2011). Indirect eavesdropping in quantum networks. In *Proceedings of the 5th ICQNM*, pages 83–88.

Schatz, D., Altheide, F., Koerfgen, H., Rossberg, M., and Schaefer, G. (2023). Virtual Private Networks in the Quantum Era: A Security in Depth Approach. In *Proceedings of SECRYPT 2023*, pages 486–494.

Schatz, D., Altheide, F., Schaefer, G., and Martius, K. (2024). Quantensichere VPN-Infrastrukturen. In *Proceedings of the 20th German IT Security Congress*.

Shor, P. W. (1997). Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.*, 26(5):1484–1509.

Standaert, F., Pereira, O., Yu, Y., Quisquater, J., Yung, M., and Oswald, E. (2010). Leakage resilient cryptography in practice. In *Towards Hardware-Intrinsic Security - Foundations and Practice*, pages 99–134. Springer.

Turan, M. S., Barker, E., Burr, W., and Chen, L. (2010). Recommendation for password-based key derivation. part 1: Storage applications. Technical report, National Institute of Standards & Technology.

Wootters, W. K. and Zurek, W. H. (1982). A single quantum cannot be cloned. *Nature*, 299(5886):802–803.