# Key Wrapping with a Fixed Permutation

Dmitry Khovratovich
University of Luxembourg

**Abstract**

We present an efficient key wrapping scheme that uses a single wide permutation and does not rely on block ciphers. The scheme is capable of wrapping keys up to 1400 bits long and processing arbitrarily long headers. Our scheme easily delivers the security level of 128 bits or higher with the master key of the same length. The permutation can be taken from the sponge hash functions such as SHA-3 (Keccak), Quark, Photon, Spongent.

We also present a simple proof of security within the concept of Deterministic Authenticated Encryption (DAE) introduced by Rogaway and Shrimpton. We extend the setting by allowing the adversary to query the permutation and following the indifferentiability setting in the security proof of the sponge construction.

**Keywords:** Key wrapping, DAE, sponge, Keccak.

## 1 Introduction

Key wrapping schemes address the problem of key management in distributed systems. Security architects often limit the lifespan of keys in order to reduce the risk of the key compromise and lessen the amount of data encrypted on a single key. Hence keys are regularly updated, and an update protocol using an insecure channel must be carefully designed. Ideally, it should be simple and efficient. Practical constraints also limit, if not forbid the use of additional mechanisms such as nonce or random number generation.

Since the early years of digital cryptography, new keys are encrypted on (*wrapped with*) a long-term (*master*) key shared between a sender and a receiver. Confidentiality of the new key must be ensured and its integrity must be protected. A key might be binded to a header, which is not encrypted (e.g., for routing purposes) but authenticated. Therefore, the key wrapping scheme is a special case of authenticated encryption with associated data (AEAD) schemes [16], where nonces and random numbers are avoided. Such a scheme may serve not only for key update, but also for a robust and misuse-resistant general purpose encryption [17].

Traditional AEAD schemes provide confidentiality (e.g., ciphertexts are indistinguishable from random strings) and data authenticity (ciphertexts constructed by an adversary must decrypt to error). They employ randomness or nonces and can be almost as efficient as regular encryption modes [12]. It has been clear that a deterministic scheme would require at least two passes over data to make each output bit depend on each input bit.

When NIST addressed the key wrapping schemes in the series of recommendations (since at least 2001), its designs, later called AES-KW and AKW [14], were highly inefficient. Moreover, those schemes carried no formal security claims or proofs. This was natural, as the first formal treatment of this problem appeared only in 2006 [17] as the concept of the Deterministic Authenticated Encryption (DAE). Still, only a few key wrapping schemes have been proposed so far in DAE or similar frameworks: SIV and S2V [17], HBS [11], BTM [10], Hash-then-Encrypt [7, 15].

None of these proposals are universal solutions. AES-KW requires 12-fold as many operations as to encrypt the same amount of data, SIV needs two keys and is not parallelizable, and the Hash-then-Encrypt template can not be scaled to a general-purpose encryption mode. Third-party analysis of these constructions is often difficult because of lengthy and complicated proofs of security. The

recently found flaw in the security proof of GCM [9] emphasizes the need for clarity and extensive third-party verification of provably secure schemes.

All these designs employ a block cipher, and the natural choice of AES limits their security level to 64 bits. To obtain the 128-bit security or higher, one would need a block cipher with a 256-bit block or larger. Except for Threefish, a component of the Skein hash function [6], no other 256-bit blockcipher enjoyed significant attention from cryptanalysts.

However, block ciphers are not the only source of good permutations. Quite recently, the hash function Keccak [3], which employs a 1600-bit permutation, has been selected as the new standard SHA-3. Since 1600 bits are often sufficient to carry the master key, the new key, and the associated data, we could restrict the use of a permutation to a single call and obtain a scheme with a reasonably short proof of security.

**Our proposal.** We present a new key wrapping scheme with a variable security level and a proof of security that is easy to verify. We call it KWF, as it is based not on a block cipher but on a fixed permutation such as those used in sponge hash functions (SHA-3/Keccak, Quark).

A wide permutation (up to 1600 bits in Keccak) easily delivers the security level of 128 bits or higher when using the key of the same length. Associated data (header) is processed with an unkeyed cryptographic hash function, possibly the same from which the permutation comes. Apart from the header processing, the scheme has no overhead over a single permutation call. We limit the message length to at max 1411 bits, but this must be sufficient for wrapping all symmetric keys and many asymmetric private keys (e.g., elliptic curve keys).

Our scheme is about as efficient as the hash function from which the permutation comes. If the associated data $H$ is processed with the same hash function, wrapping $H$ and $M$ takes roughly the same time as hashing $H||M$. We recommend using a narrow-block permutation for shorter inputs. We also note that the key length can be freely chosen.

We accompany our design with a proof of security in the refined DAE framework, where an adversary has access to the permutation as well. Here we follow the indifferentiability proof for the sponge construction [2], where the permutation is randomly chosen. This assumption establishes the security of our scheme as long as the permutation we eventually fix has no untrivial properties (which is assumed for Keccak and other sponge functions). We tried to make our proof as simple as possible to encourage its third-party verification.

Our scheme, as well as other DAE-conformant designs,is also fine for general-purpose encryption and authenticated encryption of short inputs. To emphasize this opportunity, we use the notion *message* for keys and other inputs that are encrypted by our scheme.

**Related work.** We already mentioned other designs that aim for key wrapping and deterministic authenticated encryption. NIST has published its first key wrapping scheme around 2001 (see description in [14]). AES-KW is a sort of generalized Feistel scheme, where the key to and the header are divided into 64-bit blocks, and the round function is the AES applied to two leftmost blocks. No analysis of this scheme has been published, though it is believed [17] that the security level is somewhere between 64 and 128 bits. There are several modifications to this scheme, known as KW and KWP, and a special version that uses Triple-DES.

Rogaway and Shrimpton proposed the scheme SIV, which computes MAC of the message and header with a PRF under key $K_1$ and uses it as the IV in an IV-based encryption scheme with key $K_2$ [17]. The concrete proposal invokes CMAC and CTR. The scheme SIV is provably secure in the strongest model — DAE — where the adversary can choose plaintexts and ciphertexts but is unable to distinguish the pair of encryption and decryption oracles from the pair of "random-bits" and "always error" oracles. Scheme HBS [11] and its refinement BTM [10] by Iwata and Yasuda use polynomial hash functions for MAC and a modified CTR mode. Similarly to CMAC, secondary keys are derived out of a single key by encrypting constants. HBS and BTM are provably secure in the DAE setting.

Gennaro and Halevi proposed the general template of Hash-then-Encrypt [7], which may be viewed

as weakening of SIV. Here a PRF is replaced with a hash function, which might not even be collision resistant. for instance, they showed that a composition of universal hashing and CTR mode is secure. However, the performance is gained at the cost of weakening the model. Confidentiality is achieved in the assumption of random plaintexts (RPA), where the adversary obtains two plaintexts (out of his control) and one of corresponding ciphertexts, and he has to guess which one. This "left-or-right" setting is provably weaker than DAE, but is still sufficient for key wrapping schemes where inputs have enough entropy. Osaki and Iwata [15] continued work in this direction and introduced a special class of universal hash functions which are fine to use with ECB or CBC.

**Outline of the paper.** We describe our proposal KWF in Section 2 and also recommend a set of permutations for various security levels. We immediately proceed with the security proof of KWF in Section 3. We compare KWF with other key wrapping schemes in Section 4.

# 2 Our proposal: KWF

## 2.1 Notation

For two bit strings $X$ and $Y$ of the same length, $X \oplus Y$ is their xor. For an integer $n \geq 1$, $\{0,1\}^n$ is the set of all bit strings of $n$ bits, and $\{0,1\}^{m..n}$ is the set of strings of $m$ to $n$ bits long. Also $X_{m..n}$ denotes the substring of $X$ containing bits with indices from $m$ to $n$, where the first index is 1. We write $X \xleftarrow{\$} \mathcal{X}$ for sampling an element from the set $\mathcal{X}$ uniformly at random.

## 2.2 Description of KWF

Our scheme provides an authenticated encryption of short messages and is based on a fixed $n$-bit permutation $\mathcal{F}$. Of the $n$-bit input, $k$ bits are devoted to the key $K$, $l$ bits to the associated data $H$, and $n-k-l$ to the message $M$. As the associated data needs only authentication, we map a possibly long string $H$ to an $l$-bit value with a cryptographic hash function $\mathcal{G}$.

We define scheme KWF formally as $\Pi = (\mathcal{K}, \mathcal{E}[\mathcal{F}], \mathcal{D}[\mathcal{F}])$, where:

1. $\mathcal{K} = \{0,1\}^k$ — the key space;
   $\mathcal{H} = \{0,1\}^{0..t}$ — the associated data (AD) space with $t \leq 2^k$;
   $\mathcal{M} = \{0,1\}^{1..(n-k-l-1)}$ — the message space;
   $\mathcal{C} = \{0,1\}^n$ — the ciphertext space.

2. $\mathcal{G} : \{0,1\}^{0..t} \longrightarrow \{0,1\}^l$ — collision-resistant hash function for the associated data;
   $\text{pad} : \{0,1\}^{1..(n-k-l-1)} \longrightarrow \{0,1\}^{n-l}$ — invertible padding function;
   $\mathcal{F} : \{0,1\}^n \longrightarrow \{0,1\}^n$ — fixed permutation.

3. $\mathcal{E}[\mathcal{F}] : \mathcal{K} \times \mathcal{H} \times \mathcal{M} \longrightarrow \mathcal{C}$ — encryption function (Figure 1):

$$\begin{cases} \mathcal{E}_K[\mathcal{F}](H,M) = \mathcal{F}\left(K||\mathcal{G}(H)||pad[M])\right) \oplus \left(K||0^{n-k}\right), & \text{if } H \neq \emptyset; \\ \mathcal{E}_K[\mathcal{F}](H,M) = \mathcal{F}\left(K||0^l||pad[M]\right) \oplus \left(K||0^{n-k}\right), & \text{else.} \end{cases}$$

4. $\mathcal{D}[\mathcal{F}] : \mathcal{K} \times \mathcal{H} \times \{0,1\}^n \longrightarrow \mathcal{M} \cup \{\perp\}$ — decryption function. $\mathcal{D}_K[\mathcal{F}](H,C)$ is computed as follows:
   (a) $X \leftarrow \mathcal{F}^{-1}(C \oplus \left(K||0^{n-k}\right))$.
   (b) If $X_{1..k} \neq K$, return $\perp$.
   (c) If $H = \emptyset$ and $X_{k+1..k+l} \neq 0$ return $\perp$;

(d) Else if $H \neq \emptyset$ and $X_{k+1..k+l} \neq \mathcal{G}(H)$ return $\perp$.

(e) Return $\text{pad}^{-1}(X_{k+l+1..n})$.

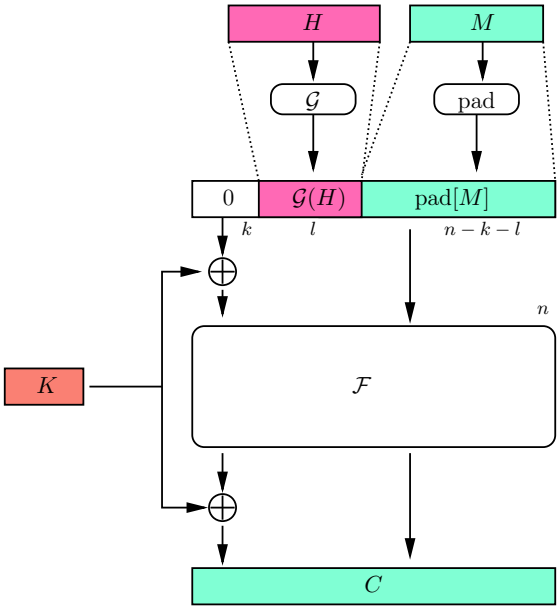


Figure 1: Our proposal: KWF.

### 2.3 Recommended parameters

In Section 3 we prove that the adversary asking at most $q$ encryption and decryption queries has the maximum advantage of

$$\frac{2q^2}{2^{n-k}} + \frac{2q}{2^k} + P^{\mathcal{G}}_{coll}(q),$$

where $P^{\mathcal{G}}_{coll}(q)$ is the success probability of the collision search for $\mathcal{G}$ with computational complexity equivalent to $q$ queries. Assuming that one query is equivalent to one call of a hash function, a collision resistant $l$-bit hash function yields the following approximation.

$$P^{\mathcal{G}}_{coll}(q) \approx \frac{q^2}{2^l}.$$

Then to get the security level of $S$ bits, the following constraints are sufficient:

$$k \geq S, \quad l \geq 2S, \quad n \geq 2S + k.$$

Therefore, for 80-bit master keys and 80-bit (padded) plaintexts we need $n \geq 320$, and for 128-bit keys and plaintexts — $n \geq 512$.

There are several families of sponge hash functions that provide suitable permutations for KWF. The Keccak family [3] uses permutations of width $n = 25 \cdot 2^l$, $l = 0..6$, with $n = 1600$ chosen for SHA-3. Keccak permutations are reasonably efficient in software and hardware, and are natural choice whenever SHA-3 is implemented on the platform. We propose to use $n = 400, 800, 1600$ and denote them as Keccak-$n$. It is natural to use the Keccak hash function also for the associated data. By Keccak/$t$ we denote the sponge hash function using the 1600-bit Keccak permutation with capacity $2t$ and rate $1600 - 2t$. The function Keccak/256 is to be standardized as SHA-3-256.

As far as we know, the key wrapping schemes are also deployed on smart cards. Hence we would like to offer a portfolio of permutations for KWF that are suitable for resource-constrained platforms.

Quark [1], Photon [8], and Spongent [5] are recently proposed families of lightweight hash functions based on the sponge construction. They use permutations from 88 to 768 bits wide and were not shown any internal weaknesses. Depending on the message length and the security level, permutations from 256 to 768 bits can be recommended for KWF. Some more details are given in Appendix A and the summary in Table 2.

Whenever SHA-2 is implemented on the platform, it also can be used as the hash function for the associated data. The most suitable for KWF are SHA-224 and SHA-256.

Some permutations are significantly faster than their inverses, e.g., the Keccak permutations. Assuming that the receiver in the key wrapping scheme is more resource-constrained, we propose to use the inverse of such permutation for encryption, and hence the forward call for the decryption.

## 3 Security of KWF

Rogaway and Shrimpton [17] were the first who introduced a single notion that amalgamates both privacy and authenticity properties of a key wrapping scheme. They called it Deterministic Authenticated Encryption (DAE). Gennaro and Halevi proposed a weaker notion of security (RPA+INT), where the plaintexts are randomly chosen and are out of adversary's control. We prove the security of KWF in a strengthened version of DAE, so our scheme is secure for general-purpose encryption where the adversary may control the plaintexts.

The security of a DAE scheme is defined as the inability to distinguish between the two worlds, where an adversary has access to two oracles [17]. One world consists of the encryption oracle $\mathcal{E}(\cdot,\cdot)$ and decryption oracle $\mathcal{D}(\cdot,\cdot)$, where the secret key is randomly chosen. The second world consists of the "random-bits" oracle $\$(\cdot,\cdot)$ and the "always-error" oracle $\perp(\cdot,\cdot)$ (Figure 2).

The *DAE-advantage* of adversary $A$ is defined as follows:

$$\mathbf{Adv}^{\mathrm{dae}}_{\Pi}(A) = \Pr\left[K \xleftarrow{\$} \mathcal{K} :\ A^{\mathcal{E}_K(\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[A^{\$(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1\right].$$

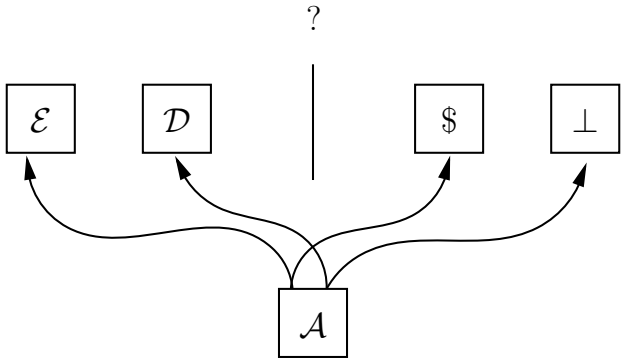


Figure 2: Indistinguishability setting for DAE.

This definition is best suitable for encryption schemes based on secure PRPs and authentication schemes based on secure PRFs. However, our concept is not based on a secure PRP, as we work with a single permutation. The provable security framework for schemes employing a single permutation typically follows [2,4] the *indifferentiability framework* [13]. Assuming no non-trivial properties of the permutation, the designers replace it with a randomly chosen permutation and prove indistinguishability from the random oracle even though the adversary can query the randomly chosen permutation as well.

We modify the indistinguishability setting of DAE accordingly, so that the adversary has access to the permutation $\mathcal{F}$, which is randomly selected, in both worlds. We call the new notion *Deterministic Authenticated Encryption with a Fixed Permutation (DAE-FP).*

We also slightly refine the definition of the "random-bit" oracle $\$(\cdot,\cdot)$ with the following motivation. Since the encryption is invertible, an ideal encryption scheme with a fixed key and associated data should be a permutation. Hence it is natural to model the oracle $\$(\cdot,\cdot)$ as an ideal cipher — a set of randomly chosen permutations indexed by a key. Here the associated data serves as a key. This model allows for an increased security level and a tighter bound, since a traditional proof of security for an encryption mode invokes a PRF and then applies the PRF-to-PRP switching lemma. This lemma limits the security level with the birthday bound, which we would like to avoid.

**Definition 1.** *Let* $\Pi = (\mathcal{K}, \mathcal{E}[\mathcal{F}], \mathcal{D}[\mathcal{F}])$ *be a DAE scheme based on permutation* $\mathcal{F}$*. The* DAE-FP advantage *of adversary* $A$ *in breaking* $\Pi$ *is computed as follows:*

$$\mathbf{Adv}^{\text{dae-fp}}_{\Pi}(A) = \Pr\left[K \xleftarrow{\$} \mathcal{K},\, \mathcal{F}(\cdot) \xleftarrow{\$} \text{Perm}(n) :\ A^{\mathcal{E}_K[\mathcal{F}](\cdot,\cdot),\mathcal{F}(\cdot),\mathcal{D}_K[\mathcal{F}](\cdot,\cdot)} \Rightarrow 1\right] - \\ - \Pr\left[\mathcal{F}(\cdot) \xleftarrow{\$} \text{Perm}(n) :\ A^{\$(\cdot,\cdot),\mathcal{F}(\cdot),\perp(\cdot,\cdot)} \Rightarrow 1\right].$$

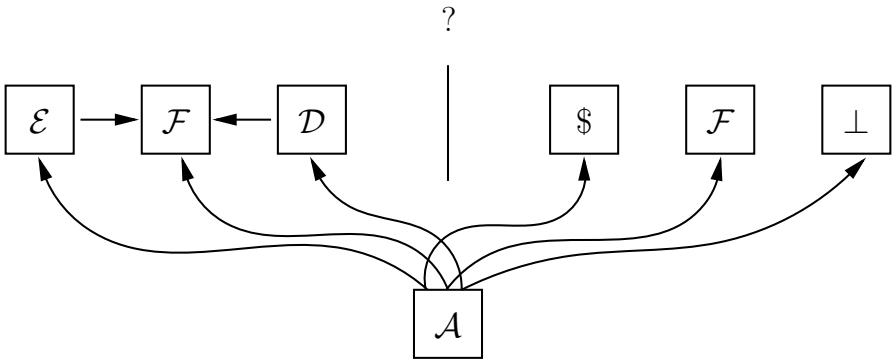


Figure 3: Indistinguishability framework for DAE-FP.

On query $(H, X)$ the oracle $\$(\cdot,\cdot)$ returns a random string of length $n$ so that it is a permutation (bijective function) for every $H$. The set of all permutations over $\{0,1\}^n$ is denoted by $\text{Perm}(n)$. The $\perp(\cdot,\cdot)$ oracle always returns $\perp$ (error). We exclude trivial wins: the adversary shall not ask $(H, Y)$ of its right oracle if some previous left oracle query of $(H, X)$ returned $Y$ and vice versa. Without loss of generality, the adversary does not repeat a query and does not ask left queries outside of $\mathcal{H} \times \mathcal{M}$. Here and in the further text we implicitly assume that $\mathcal{F}^{-1}$ is available together with $\mathcal{F}$.

Let us express the maximum advantage as a function of the number of allowed queries:

$$\mathbf{Adv}^{\text{dae-fp}}_{\Pi}(q) \stackrel{\text{def}}{=} \max_A \mathbf{Adv}^{\text{dae-fp}}_{\Pi}(A),$$

where we take maximum over all adversaries asking at maximum $q$ queries.

**Theorem 1.** *The DAE-FP advantage of an adversary attacking KWF and asking the total of at most* $q$ *queries to all oracles is bounded as follows:*

$$\boldsymbol{Adv}^{dae\text{-}fp}_{\Pi}(q) \le \frac{2q^2}{2^{n-k}} + \frac{2q}{2^k} + P^{\mathcal{G}}_{coll}(q),$$

*where* $P^{G}_{coll}(q)$ *is the probability of obtaining a collision for the hash function* $\mathcal{G}$ *after* $q$ *queries or their computational equivalent.*

*Proof.* We split this expression in two following the approach in [17]:

$$\Pr\left[K \xleftarrow{\$} \mathcal{K},\, \mathcal{F}(\cdot) \xleftarrow{\$} \text{Perm}(n) :\ A^{\mathcal{E}_K[\mathcal{F}],\mathcal{F},\mathcal{D}_K[\mathcal{F}]} \Rightarrow 1\right] - \Pr\left[\mathcal{F}(\cdot) \xleftarrow{\$} \text{Perm}(n) :\ A^{\$,\mathcal{F},\perp} \Rightarrow 1\right] = \\ = \underbrace{\Pr\left[A^{\mathcal{E}_K[\mathcal{F}],\mathcal{F},\mathcal{D}_K[\mathcal{F}]} \Rightarrow 1\right] - \Pr\left[A^{\mathcal{E}_K,\mathcal{F},\perp} \Rightarrow 1\right]}_{p_1} + \underbrace{\Pr\left[A^{\mathcal{E}_K,\mathcal{F},\perp} \Rightarrow 1\right] - \Pr\left[A^{\$,\mathcal{F},\perp} \Rightarrow 1\right]}_{p_2}.$$

### 3.1 Bounding $p_1$ (authenticity proof)

Consider

$$p_1 = \Pr\left[A^{\mathcal{E}_K[\mathcal{F}],\mathcal{F},\mathcal{D}_K[\mathcal{F}]} \Rightarrow 1\right] - \Pr\left[A^{\mathcal{E}_K,\mathcal{F},\perp} \Rightarrow 1\right],$$

where $K$ and $\mathcal{F}$ are randomly chosen.

We assume without loss of generality that $A$ halts and outputs 1 whenever the right oracle returns $M \neq \perp$. Prior to this event, both oracle triplets behave identically as $(\mathcal{E}_K, \mathcal{F}, \perp)$. Therefore, $p_1$ is bounded by the probability that $A$ asks a right-oracle query $(H, Y)$ so that $\mathcal{D}_K(H, Y) \neq \perp$.

Let us denote the set of ciphertexts obtained prior to this query by $\overline{\mathcal{C}}$, the set of $\mathcal{F}$ responses and $\mathcal{F}^{-1}$ queries by $\mathcal{F}_o$, and of $\mathcal{F}$ queries and $\mathcal{F}^{-1}$ responses by $\mathcal{F}_i$.

By definition, the adversary is unable to use a pair $(H, Y)$ where $Y$ has been a response $Y = \mathcal{E}_K(H, X)$ for some $X$. Hence either $Y \notin \overline{\mathcal{C}}$, or $Y = \mathcal{E}_K(H', X), H' \neq H$. In the latter case the ciphertext decrypts to $\mathcal{G}(H')||X$, so the decryption returns error unless $H$ and $H'$ form a collision pair for $\mathcal{G}$. We denote the success rate of the collision search for $\mathcal{G}$ given the computational equivalent of $q$ queries[1] by $P^{\mathcal{G}}_{\text{coll}}(q)$.

If either of headers is empty, authenticity is violated if the other header is mapped by $\mathcal{G}$ to zero. We assume that such an input is infeasible to find with the complexity lower than the collision search.

If $Y \notin \overline{\mathcal{C}}$, then

$$\Pr\left[\mathcal{D}_K(H, Y) \neq \perp\right] \leq \Pr\left[\mathcal{D}_K(H, Y) \neq \perp \mid (Y \oplus K) \notin \mathcal{F}_o\right] + \\ + \Pr\left[\mathcal{D}_K(H, Y) \neq \perp \mid (Y \oplus K) \in \mathcal{F}_o\right].$$

- $(Y \oplus K) \notin \mathcal{F}_o$. Then the permutation $\mathcal{F}^{-1}$ is asked with a fresh query, so its output is uniformly distributed along previously unallocated values. The decryption returns error if

  $$F^{-1}(Y \oplus K)_{1..k+l} \neq K||G(H).$$

  Hence at maximum $2^{n-k-l}$ values pass this condition. As at minimum $2^n - q$ values remain unassigned, we obtain

  $$\Pr\left[\mathcal{D}_K(H, Y) \neq \perp \mid (Y \oplus K) \notin \mathcal{F}_o\right] \leq \frac{2^{n-k-l}}{2^n - q}. \tag{1}$$

- $(Y \oplus K) \in \mathcal{F}_o$. Hence the decryption oracle ask the permutation with a query that is not fresh. Then the decryption returns $\perp$ if

  $$\mathcal{F}^{-1}(Y \oplus K) \neq K||Z$$

  for any $Z$. We say that the *input clash* (IC) occurs, if $(K||Z) \in \mathcal{F}_i$ for some $Z$. Hence without the input clash the decryption error is guaranteed:

  $$\Pr\left[\mathcal{D}_K(H, Y) \neq \perp \mid (Y \oplus K) \in \mathcal{F}_o, \text{ no IC occurred}\right] = 0.$$

Finally,

$$\Pr\left[\mathcal{D}_K(H, Y) \neq \perp \mid \text{ no IC occurred}\right] \leq \frac{2^{n-k-l}}{2^n - q}.$$

and

$$p_1 = P^{\mathcal{G}}_{\text{coll}}(q) + \left[1 - \left(1 - \frac{2^{n-k-l}}{2^n - q}\right)^q\right] + \Pr(IC; q) \leq P^{\mathcal{G}}_{\text{coll}}(q) + \frac{q}{2^{k+l-1}} + \Pr(IC; q).$$

[1] We could introduce a separate resource unit, e.g., time $t$, to evaluate the probability of getting a collision. However, we assume that in practice it is easy to convert the number of queries to the encryption/decryption functions to the resources needed for the collision search.

It remains to bound the probability $\Pr(IC; q)$ of getting the input clash after $q$ queries. Here either the adversary tries to guess the key in $\mathcal{F}$ queries or hopes to obtain it as a prefix in $\mathcal{F}^{-1}$ responses. In the worst case, when all the $q$ prefixes are different, the probability of having $K$ among them is bounded by $q/2^k$, so we have the following bound on the input clash:

$$\Pr(IC; q) \leq \frac{q}{2^k}. \tag{2}$$

This gives the final bound on $p_1$:

$$p_1 \leq P^{\mathcal{G}}_{\text{coll}}(q) + \frac{q}{2^{k+l-1}} + \frac{q}{2^k} \approx P^{\mathcal{G}}_{\text{coll}}(q) + \frac{q}{2^k}.$$

### 3.2 Bounding $p_2$ (privacy proof)

Recall that

$$p_2 = \Pr\left[A^{\mathcal{E}_K[\mathcal{F}],\mathcal{F},\perp} \Rightarrow 1\right] - \Pr\left[A^{\$,\mathcal{F},\perp} \Rightarrow 1\right],$$

where $K$ and $\mathcal{F}$ are chosen randomly.

We can drop the oracle $\perp$ simply by considering the adversary $B$ that has access to the left and center oracles only and runs $A$. She transfers queries of $A$ directly to the oracles and returns $\perp$ to all queries by $A$ to the right oracle. Hence

$$p_2 \leq \Pr\left[B^{\mathcal{E}_K[\mathcal{F}],\mathcal{F}} \Rightarrow 1\right] - \Pr\left[B^{\$,\mathcal{F}} \Rightarrow 1\right].$$

In the further text we show that in the absence of two events, so called the output clash and the oracle repetition, oracles $\mathcal{E}_K[\mathcal{F}]$ and \$ produce identically distributed results and hence are indistinguishable. The clashes and some remarks on how they can be exploited are depicted in Figure 4.

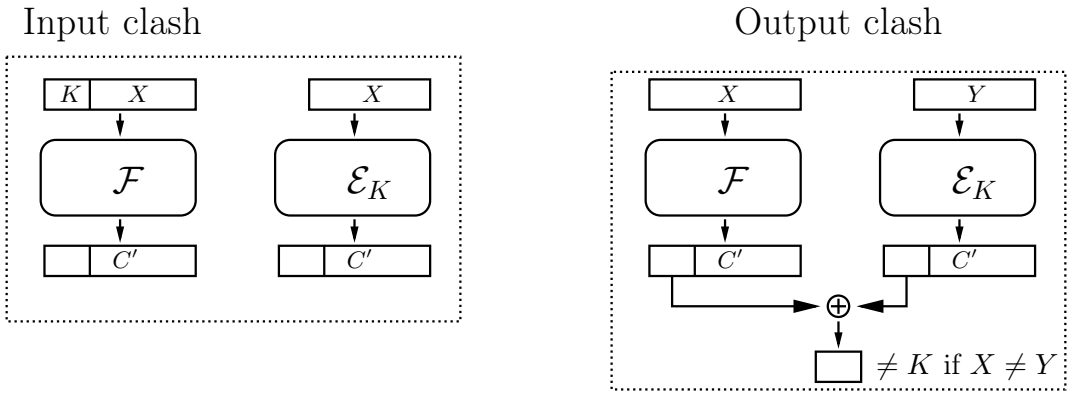


Figure 4: Detection and exploit of the clashes. In case of input clash we encrypt $(n-k)$-bit suffix $X$ of the permutation query $K||X$ and detect suffix collision in outputs. In case of output clash some ciphertext prefixes are forbidden, which allow to distinguish the encryption from the ideal cipher.

Let us go to the details. Consider the query $(H, X)$ to the encryption oracle $\mathcal{E}$ or \$. Denote by $\overline{C}$ the set of encryption oracle responses obtained beforehand. We are also interested in the last $(n-k)$ bits of ciphertext, which are unaffected by the key addition. We denote $C|_{k+1..n}$ by $\widehat{C}$ and use the same notation for $(n-k)$-bit suffixes of $\mathcal{F}_o$ denoted by $\widehat{\mathcal{F}_o}$.

Let us say that the encryption oracle response $C = \mathcal{E}_K(H, X)$ causes the *output clash* if $\widehat{C} \in \widehat{\mathcal{F}_o}$, i.e. the ciphertext collides with one of stored permutation outputs on the last $n-k$ bits. Let $\mathcal{V}$ be the set of the ciphertexts that did not occur in previous responses and do not cause the output clash:

$$\mathcal{V} \stackrel{\text{def}}{=} \left[\{0,1\}^k \times \left(\{0,1\}^{n-k} \setminus \widehat{\mathcal{F}_o}\right)\right] \setminus \overline{E}.$$

**Lemma 1.** *If the input clash did not occur, then all responses $\mathcal{E}_K(H, X) \in \mathcal{V}$ are equiprobable.*

*Proof.* Since there was no input clash, the input $K||\mathcal{G}(H)||pad[X]$ has not been queried to the permutation. Hence, it is fresh, and its output is uniformly taken from $\{0,1\}^n \setminus \overline{\mathcal{C}}$. It remains to remove from this set the values that cause the output clash, i.e. the set $\{0,1\}^k \times \left(\{0,1\}^{n-k} \setminus \widehat{\mathcal{F}_o}\right)$. This concludes the proof. □

The ideal cipher has a different distribution, as its outputs may collide for distinct $H$. Let us say that the *oracle repetition* (OR) occurs if $\$(H, X) = \$(H', X')$ for some previously queried $(H', X')$. If we exclude the oracle repetition and output clash events, the distribution will be the same:

$$\$(H, X) \xleftarrow{\$ \,|\, \text{no OR, no OC}} \mathcal{V}.$$

Combining with Lemma 1, we conclude that if no input clash, no output clash, and no oracle repetition occurs, the encryption oracles produce identically distributed responses and are indistinguishable. Therefore,

$$p_2 \le \Pr(IC; q) + \Pr(\text{OC for } \mathcal{E} \,|\, \text{no IC}; q) + \Pr(\text{OC for } \$(\cdot,\cdot); q) + \Pr(OR). \tag{3}$$

**Output clash bound.** Provided no input clash, a query $\mathcal{E}(H, X)$ yields a fresh query to $\mathcal{F}$. Hence the output is uniformly distributed among at least $2^n - q$ previously unassigned values, of which at maximum $q \times 2^k$ cause the output clash. Therefore, assuming $q \le 2^k$

$$\Pr\left(\widehat{\mathcal{E}(H, X)} \in \widehat{\mathcal{F}_o}\right) \le \frac{q2^k}{2^n - q} \approx \frac{q}{2^{n-k}}.$$

Thus we have the following bound:

$$\Pr(\text{OC for } \mathcal{E} \,|\, \text{no IC}; q) \le 1 - \left(1 - \frac{q}{2^{n-k}}\right)^q \approx \frac{q^2}{2^{n-k}}. \tag{4}$$

The bound for the ideal cipher is the same:

$$\Pr\left(\widehat{\$(H, X)} \in \widehat{\mathcal{F}_o}\right) \le \frac{q \cdot 2^k}{2^n - q} \implies \Pr(\text{OC for } \$(\cdot,\cdot); q) \le \frac{q^2}{2^{n-k}}. \tag{5}$$

**Oracle repetition bound.** We calculate the probability of the event that during $q$ queries to the PRP-oracle \$ there is no collision in outputs. Consider $i$-th query. The oracle chooses its output uniformly out of at least $(2^n - q)$ possibilities, of which at maximum $q$ cause a collision. Hence

$$\Pr(OR;\, q) = 1 - \prod_{i=1}^{q} \left(\frac{2^n - 2q}{2^n - q}\right) = 1 - \left(1 - \frac{q}{2^n - q}\right)^q \le 1 - e^{-\frac{2q^2}{2^n}} \approx \frac{2q^2}{2^n} \ll \frac{q^2}{2^{n-k}} \tag{6}$$

We substitute Equations (2), (4), (5), and (6) to Equation (3) and obtain the final bound

$$p_2 \le \frac{2q^2}{2^{n-k}} + \frac{q}{2^k}.$$

This concludes the proof of Theorem 1. □

# 4 Conclusion

We have described a new key wrapping scheme — KWF. It is based on a single fixed permutation, with a lot of candidate permutations available in sponge designs: Keccak/SHA-3, Quark, Spongent, Photon. Though keys to be wrapped are limited to the length of 1411 bits, our scheme still provides a simple and efficient key update protocol for most of symmetric and several asymmetric cryptosystems. It can also bind the associated data to the message, and is able to preprocess it without the master key. The ciphertext length is equal to the permutation width $n$, while the master key length $k$ can vary.

The scheme is provably secure in the refined concept of DAE, where we add the adversarial access to the permutation. Assuming no weakness in the permutation and a collision resistant hash function for the associated data, the violation of DAE property is unlikely for the number of queries $q \leq 2^k$. Hence the security level of 128 bits is easy to deliver, which is not the case for other key wrapping schemes using AES.

A scalable version of our scheme, which would process arbitrarily long messages and remain simple and secure, is an object for the future work.

We compare KWF to other key wrapping schemes in Table 1. The overhead is usually defined as the ratio of extra blockcipher calls compared to the sole encryption of the same data; we simply divide the permutation width by the message length. The expansion is the the number of extra bits compared to the ciphertext of the same data. We also compare the need of block cipher or a hash function, the ability to preprocess the header in advance, and the ability to deliver 128-bit security with recommended parameters (usually AES for other schemes).

We conclude that KWF has moderate overhead and expansion, enjoys the security proof in the strongest model, and use only a hash function or its part whereas the others need a blockcipher. KWF is the only scheme that offers 128-bit security by default, while the other schemes need a block cipher with at least 256-bit block for this purpose.

| | KWF | HBS | AES-KW | SIV | Hash-then-E |
|---|---|---|---|---|---|
| Message length | 0..1411 | Arbitrary | | | |
| Overhead | 1.2–1.5 | 2 | 12 | 2 | $1+\varepsilon$ |
| Expansion | 192–384 | 128 | $\|H\| + 64$ | 128 | 128 |
| Security proof | DAE-FP | DAE | No | DAE | RPA |
| Block cipher | No | Yes | Yes | Yes | Yes |
| Hash function | Yes | Maybe | No | Yes | Yes |
| Preprocess header | Yes | No | No | Yes | Yes |
| 128-bit security | Yes | Not with AES | No | Not with AES | |

Table 1: Comparison of KWF with other key wrapping schemes

# References


[1] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and María Naya-Plasencia. Quark: A lightweight hash. In *CHES'10*, volume 6225 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2010. `https://131002.net/quark/quark_full.pdf`.

[2] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In *EUROCRYPT'08*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 2008.

[3] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. The Keccak reference, version 3.0, 2011. http://keccak.noekeon.org/Keccak-reference-3.0.pdf.

[4] Rishiraj Bhattacharyya, Avradip Mandal, and Mridul Nandi. Security analysis of the mode of JH hash function. In *FSE'10*, volume 6147 of *Lecture Notes in Computer Science*, pages 168–191. Springer, 2010.

[5] Andrey Bogdanov, Miroslav Knezevic, Gregor Leander, Deniz Toz, Kerem Varici, and Ingrid Verbauwhede. Spongent: A lightweight hash function. In *CHES'11*, volume 6917 of *Lecture Notes in Computer Science*, pages 312–325. Springer, 2011.

[6] Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, and Jesse Walker. The Skein hash function family, http://www.skein-hash.info/sites/default/files/skein1.3.pdf. Submission to NIST (Round 3), 2010.

[7] Rosario Gennaro and Shai Halevi. More on key wrapping. In *SAC'09*, volume 5867 of *Lecture Notes in Computer Science*, pages 53–70. Springer, 2009.

[8] Jian Guo, Thomas Peyrin, and Axel Poschmann. The PHOTON family of lightweight hash functions. In *CRYPTO'11*, volume 6841 of *Lecture Notes in Computer Science*, pages 222–239. Springer, 2011. https://sites.google.com/site/photonhashfunction/downloads/photon-full.pdf?attredirects=0.

[9] Tetsu Iwata, Keisuke Ohashi, and Kazuhiko Minematsu. Breaking and repairing GCM security proofs. In *CRYPTO'12*, volume 7417 of *Lecture Notes in Computer Science*, pages 31–49. Springer, 2012.

[10] Tetsu Iwata and Kan Yasuda. BTM: A single-key, inverse-cipher-free mode for deterministic authenticated encryption. In *SAC'09*, volume 5867 of *Lecture Notes in Computer Science*, pages 313–330. Springer, 2009.

[11] Tetsu Iwata and Kan Yasuda. HBS: A single-key mode of operation for deterministic authenticated encryption. In *FSE'09*, volume 5665 of *Lecture Notes in Computer Science*, pages 394–415. Springer, 2009.

[12] Ted Krovetz and Phillip Rogaway. The software performance of authenticated-encryption modes. In *FSE'11*, volume 6733 of *Lecture Notes in Computer Science*, pages 306–327. Springer, 2011.

[13] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In *TCC'04*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2004.

[14] NIST. Special publication 800-38f: Recommendation for block cipher modes of operation: Methods for key wrapping, 2008. http://csrc.nist.gov/publications/drafts/800-38F/Draft-SP800-38F_Aug2011.pdf.

[15] Yasushi Osaki and Tetsu Iwata. Further more on key wrapping. *IEICE Transactions*, 95-A(1):8–20, 2012. Also published at SKEW 2011: http://skew2011.mat.dtu.dk/proceedings/Further%20More%20on%20Key%20Wrapping.pdf.

[16] Phillip Rogaway. Authenticated-encryption with associated-data. In *ACM Conference on Computer and Communications Security'02*, pages 98–107, 2002.

[17] Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In *EUROCRYPT'06*, volume 4004 of *Lecture Notes in Computer Science*, pages 373–390. Springer, 2006.

# A Lightweight permutations for KWF

Here we recommend for KWF some particular permutations taken from the lightweight hash function families: Quark, Photon, Spongent.

The Quark family [1] offers compact and low-power hash functions best suitable for RFID technology and other resource-constrained platforms. It uses permutations of width $n = 136, 176, 256$. Here $n = 256$ (s-Quark) is a good candidate for KWF operating on 64-bit keys and plaintexts, and u- and d-Quark ($n = 136, 176$) can be chosen as the hash function $\mathcal{G}$ with the 64-bit and 80-bit level of collision resistance, respectively. We denote Quark permutations of width $n$ by Quark-$n$.

The Photon family [8] is an AES-based lightweight hash function design with permutations of width 100, 144, 196, 256, 288. We denote its permutations of width $n$ by Photon-$n$ and recommend using Photon-288 for applications with 64-bit level security. The hash functions Photon/$n$ offer $n/2$-bit of security for collision resistance and are suitable for processing associated data.

The Spongent family [5] is another lightweight hash function design with permutations of width from 88 to 768 bits, with all widths multiple of 4 bits available. Its permutations Spongent-$n$ can be used for all security levels, with a particular choice optimized for the input length.

| Hash function for AD | Message length | Permutation |
|---|---|---|
| Security level and key length: 64 | | |
| Keccak/128, u-Quark, Photon/128 | 0..63 | Keccak-400, Quark-256, Photon-288, Spongent-256 |
| Keccak/128, u-Quark, Photon/128 | 64..212 | Keccak-400, Spongent-400 |
| Keccak/128, u-Quark, Photon/128 | 213..612 | Keccak-800 |
| Keccak/128, u-Quark, Photon/128 | 613..1412 | Keccak-1600 |
| Security level and key length: 80 | | |
| Keccak/160, d-Quark, Photon/160 | 0..160 | Keccak-400, Spongent-400 |
| Keccak/160, d-Quark, Photon/160 | 161..560 | Keccak-800 |
| Keccak/160, d-Quark, Photon/160 | 561..1360 | Keccak-1600 |
| Security level and key length: 112 | | |
| SHA-224, Keccak/224, s-Quark | 0..464 | Keccak-800 |
| SHA-224, Keccak/224, s-Quark | 465..1264 | Keccak-1600 |
| Security level and key length: 128 | | |
| SHA-256, Keccak/256 | 0..415 | Keccak-800 |
| SHA-256, Keccak/256 | 416..1215 | Keccak-1600 |

Table 2: Recommended parameters for scheme KWF. All the numbers are bit lengths.