SP800-227
Recommendations for KEMs

Gorjan Alagić
UMD / NIST

Background

Why SP 800-227?

- *prior to PQC:* KEMs relatively uncommon;
- *PQC:* everything is a KEM!

- idea for SP during writing of FIPS 203 (ML-KEM)
  [as a place to put all the stuff that wasn't specific to ML-KEM.]
- *later:* need for hybrid PQC key-establishment guidance;

Released: January 7th

This workshop: February 25th

Comments due: March 7th

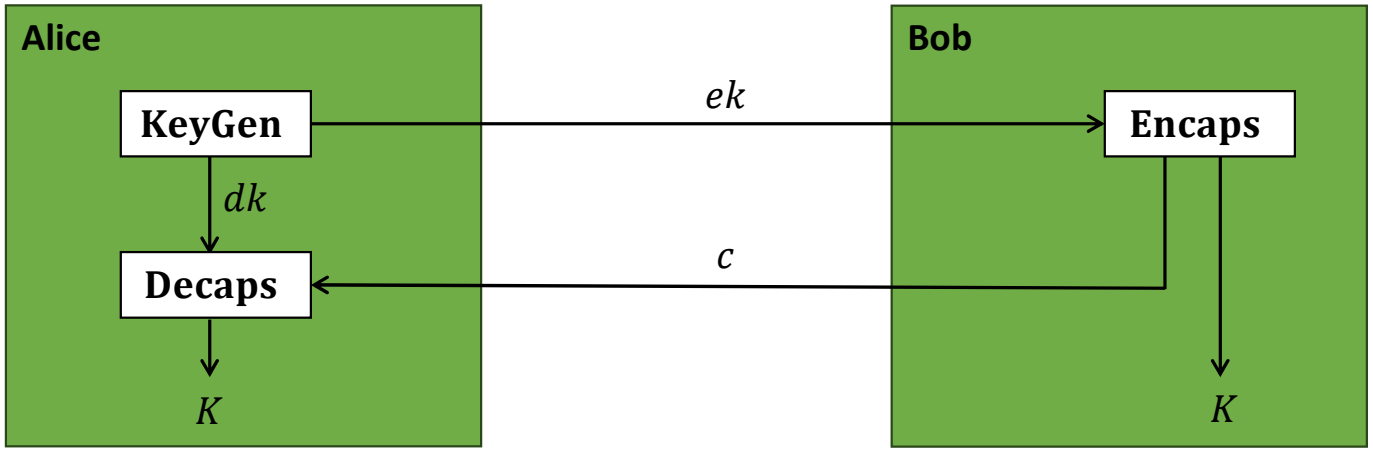
Background

What is a KEM? Key-establishment mechanism.

Consists of three algorithms:

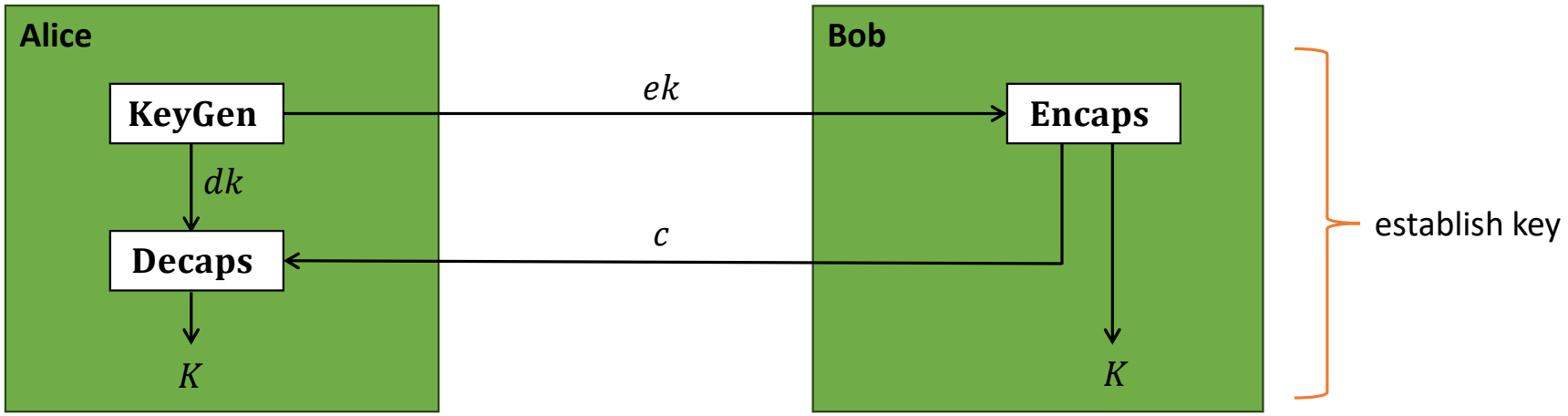
1. $(ek, dk) \leftarrow \mathbf{KeyGen}()$; key generation
2. $(K, c) \leftarrow \mathbf{Encaps}(ek)$; encapsulation
3. $K \leftarrow \mathbf{Decaps}(dk, c)$; decapsulation

Typical usage: key establishment.



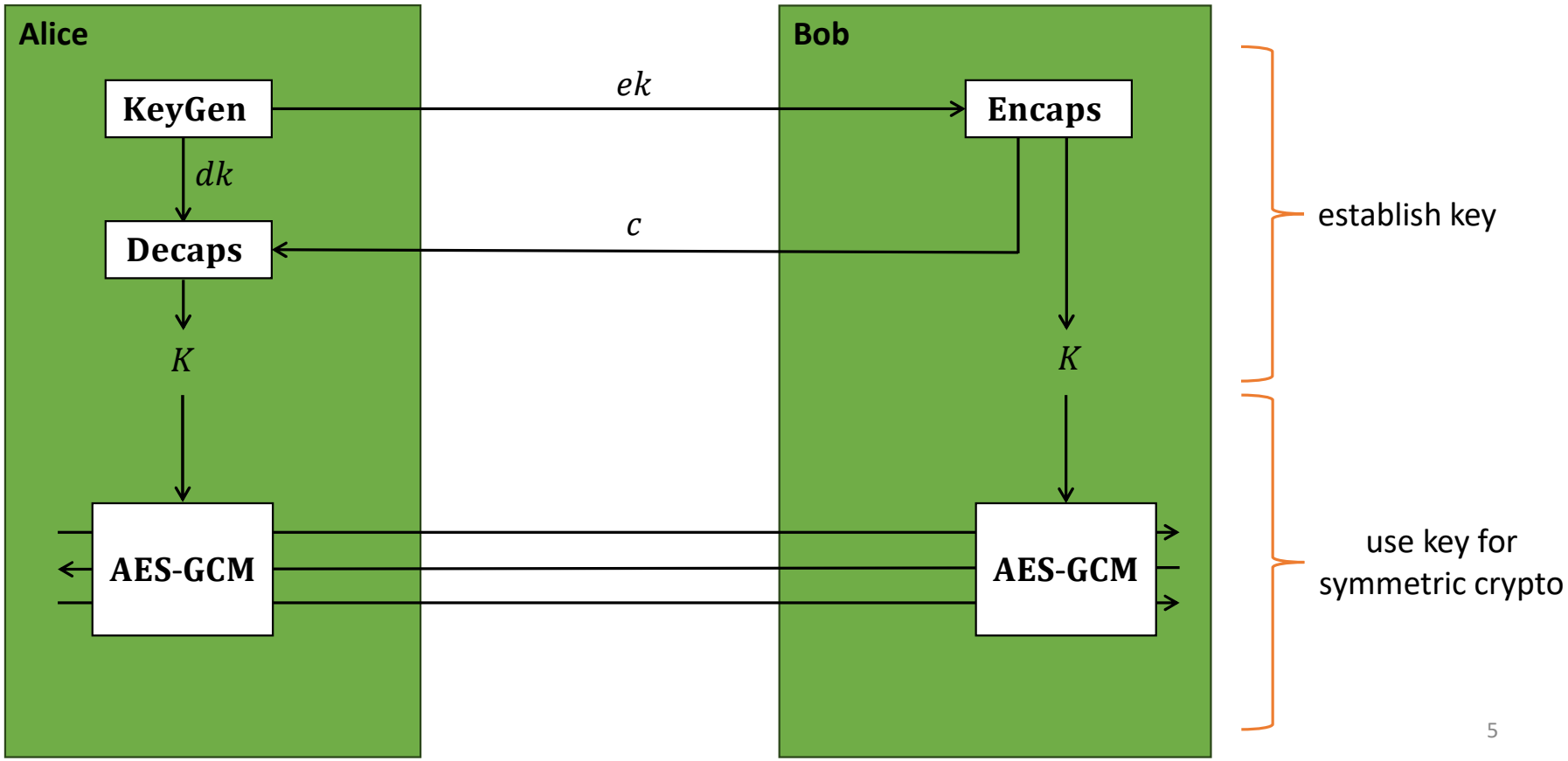
Background

What is a KEM? Key-establishment mechanism.



Background

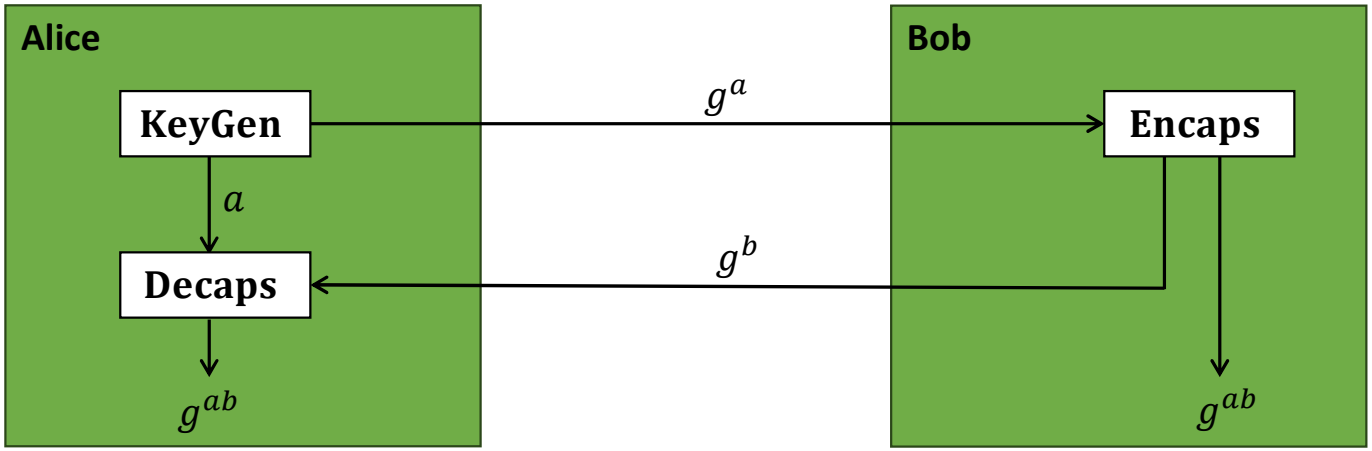
What is a KEM? Key-establishment mechanism.



Background

Examples of KEMs in NIST standards.

1. RSA "SVE" from SP800-56Br2 (essentially, PKE key transport.)
2. ML-KEM from FIPS 203 (variant of Kyber).
3. ECDH from SP800-56Ar3, like this:



SP 800-227: overview

The three life stages of a KEM.

1. **Theoretical** (Section 3): a collection of mathematical functions.

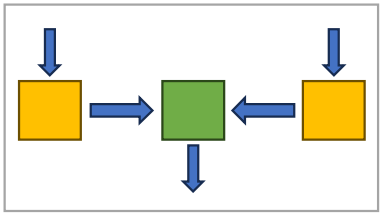
$$\text{KeyGen}: \{0,1\}^n \to \{0,1\}^m \times \{0,1\}^\ell$$
$$\ldots$$
$$\text{Encaps}: \{0,1\}^\ell \to \{0,1\}^k \times \{0,1\}^t$$
$$\ldots$$
$$\text{Decaps}: \{0,1\}^m \times \{0,1\}^t \to \{0,1\}^k$$
$$\ldots$$

2. **Implemented** (Section 4): a collection of computer programs.

```
int crypto_kem_keypair_derand(uint8_t *pk,
                              uint8_t *sk,
                              const uint8_t *coins)
{
  indcpa_keypair_derand(pk, sk, coins);
  memcpy(sk+KYBER_INDCPA_SECRETKEYBYTES, pk, KYBER_PUBLICKEYBYTES);
  hash_h(sk+KYBER_SECRETKEYBYTES-2*KYBER_SYMBYTES, pk, KYBER_PUBLICKEYBYTES);
  /* Value z for pseudo-random output on reject */
  memcpy(sk+KYBER_SECRETKEYBYTES-KYBER_SYMBYTES, coins+KYBER_SYMBYTES, KYBER_SYMBYTES);
```

...

3. **Deployed** (Section 5): software inside a larger protocol (and/or a real device).



SP 800-227: overview

$$\text{KeyGen}: \{0,1\}^n \rightarrow \{0,1\}^m \times \{0,1\}^\ell$$
$$\dots$$
$$\text{Encaps}: \{0,1\}^\ell \rightarrow \{0,1\}^k \times \{0,1\}^t$$
$$\dots$$
$$\text{Decaps}: \{0,1\}^m \times \{0,1\}^t \rightarrow \{0,1\}^k$$
$$\dots$$

KEM Stage 1. Theoretical (Section 3): a collection of mathematical functions.

At this stage, can talk about things like:

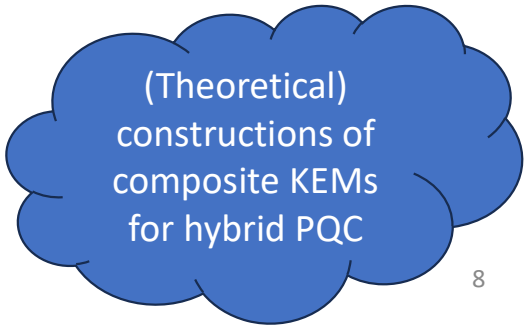
- Correctness (and decryption failure rates (DFR));
- Parameter sets;
- Theoretical security (IND-CPA, IND-CCA);
- Proofs (e.g., in models like ROM/QROM).

Our guidance at-a-glance:

- Use correct KEMs :) control DFR appropriately for security strength;
- Aim for IND-CCA whenever possible.

Composite KEMs (Sec 5.5):

- Use **approved** combiners
- Aim for IND-CCA-preserving combiners.



SP 800-227: overview

KEM Stage 2. Implemented (Section 4): a collection of computer programs.

```
int crypto_kem_keypair_derand(uint8_t *pk,
                              uint8_t *sk,
                              const uint8_t *coins)
{
  indcpa_keypair_derand(pk, sk, coins);
  memcpy(sk+KYBER_INDCPA_SECRETKEYBYTES, pk, KYBER_PUBLICKEYBYTES);
  hash_h(sk+KYBER_SECRETKEYBYTES-2*KYBER_SYMBYTES, pk, KYBER_PUBLICKEYBYTES);
  /* Value z for pseudo-random output on reject */
  memcpy(sk+KYBER_SECRETKEYBYTES-KYBER_SYMBYTES, coins+KYBER_SYMBYTES, KYBER_SYMBYTES);
                                                    ...

                                                    ...
```

https://github.com/pq-crystals

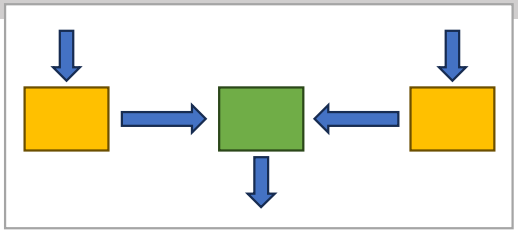
At this stage, can talk about things like:

- Cryptographic module boundaries;
- Managing data (input checking, import/export, assurances);
- Side-channel protection;
- Validation.

Our guidance at-a-glance:

- Follow NIST specs and IGs, get validated;
- Use **approved** components (e.g., RBGs) with appropriate strength (match KEM params);
- Destroy data once no longer needed;
- Get assurances of validity for imports;
- Alternative key formats (e.g., **KeyGen** seeds) are fine.

SP 800-227: overview



KEM Stage 3. Deployed (Section 5): software inside a larger protocol (and/or a real device).

At this stage, can talk about things like:

- Key derivation and key confirmation (KC);
- Hybrids (multi-algorithm schemes);
- Authentication;
- Hardware (e.g., device and channel) security.

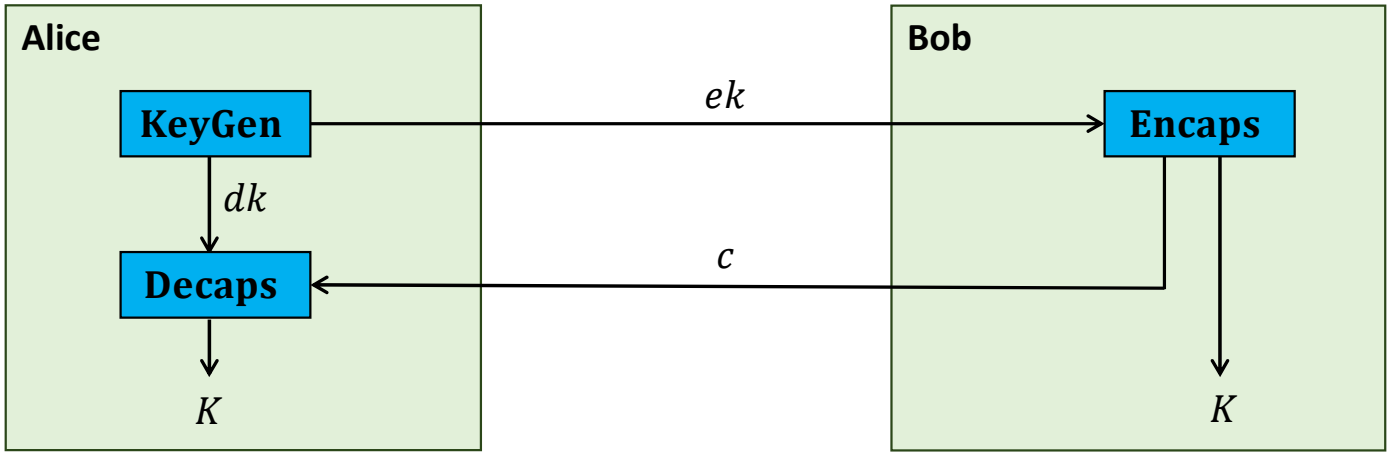
Our guidance at-a-glance:

- Do KC as in 56A and 56B (adapted to KEMs appropriately);
- Hybrids: use **approved** combiners, aim for IND-CCA-preserving;
- Select parameter sets appropriately for application;
- Do the right things for hardware security.

SP 800-227: hybrids

Hybrid PQC. (Or: multi-algorithm / composite schemes).

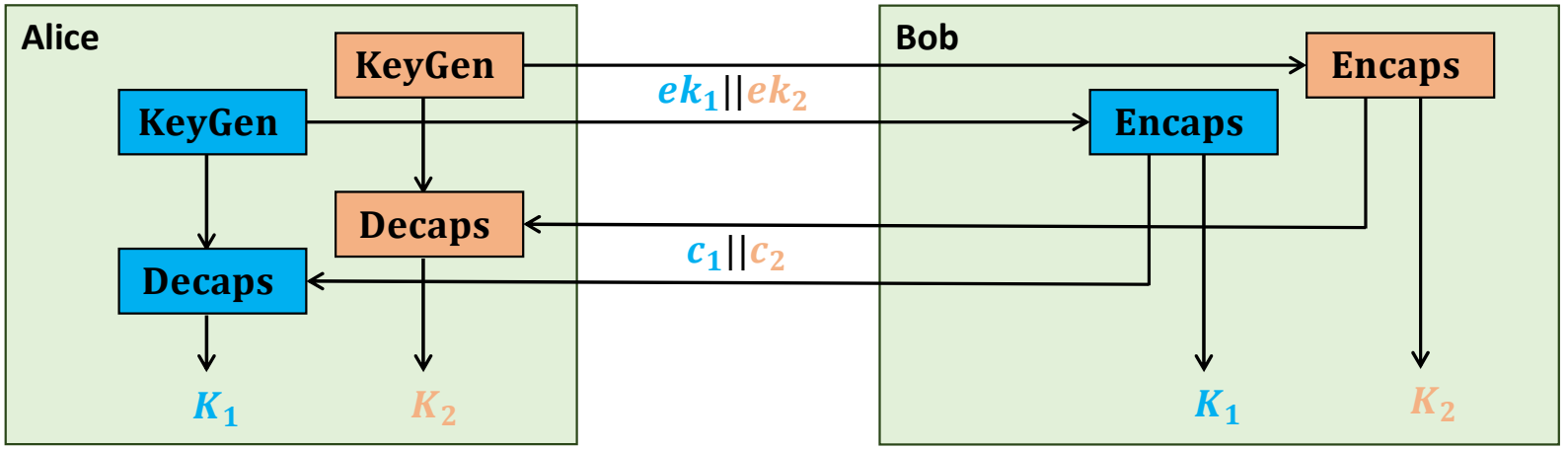
- *motivation:* new stuff sometimes gets broken.
- *solution:* combo stuff so adversary has to break **both** the new stuff and the old stuff.



SP 800-227: hybrids

Hybrid PQC. (Or: multi-algorithm / composite schemes).

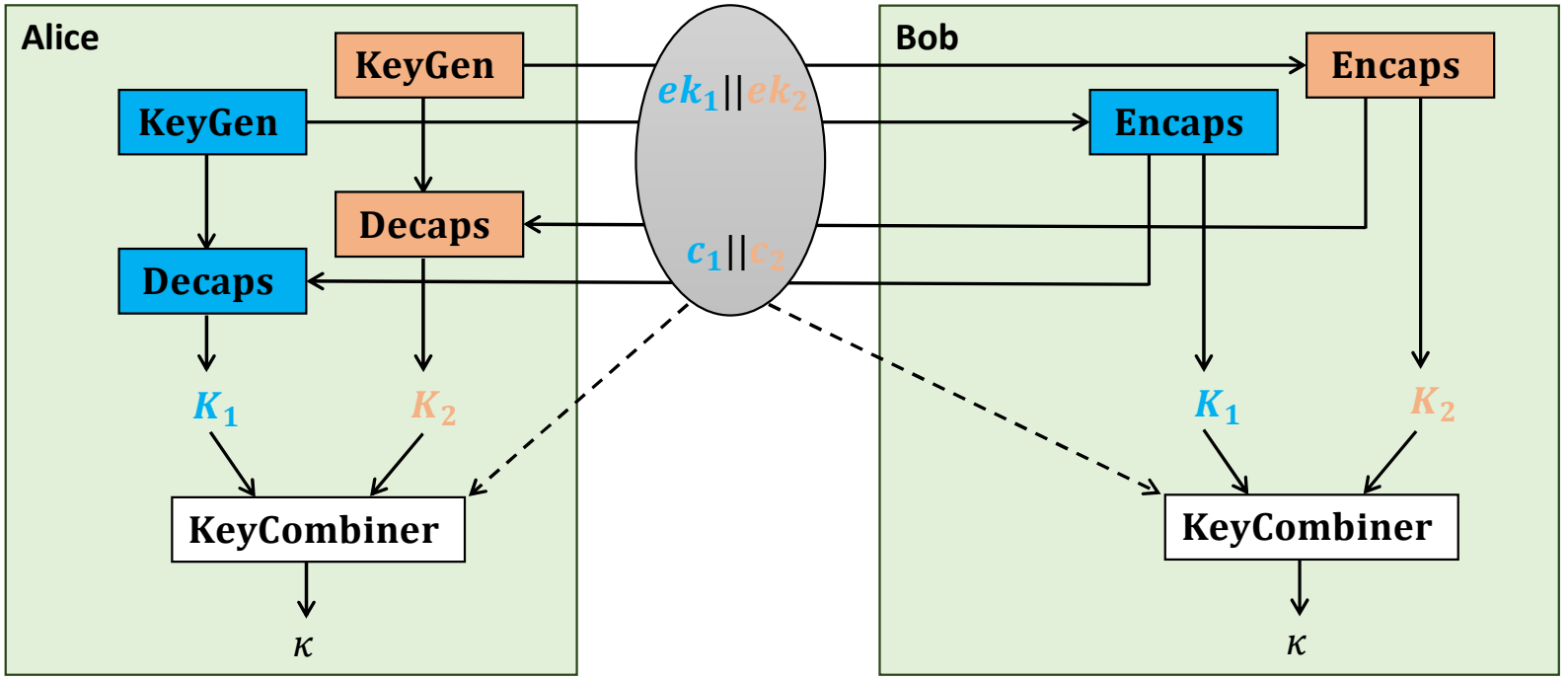
- *motivation:* new stuff sometimes gets broken.
- *solution:* combo stuff so adversary has to break **both** the **new stuff** and the **old stuff**.



SP 800-227: hybrids

Hybrid PQC. (Or: multi-algorithm / composite schemes).

- *motivation:* new stuff sometimes gets broken.
- *solution:* combo stuff so adversary has to break **both** the **new stuff** and the **old stuff**.



SP 800-227: hybrids

Hybrid KE. How to combine keys? Three approved methods.

Methods 1 and 2: Use SP 800-56C key derivation.

1.) One-step:

e.g., SHA3-256

optional but recommended (CCA security)

$$\kappa \leftarrow \mathbf{KDF}(K_1||K_2||ek_1||ek_2||c_1||c_2||\text{OtherInput})$$

any order is ok

at least one must be approved

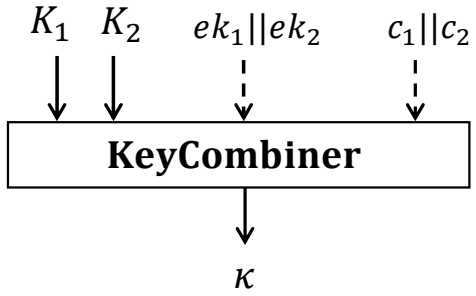
domain separators can go here

2.) Two-step:

e.g., HMAC

$$\kappa \leftarrow \mathbf{Expand}(\mathbf{Extract}(\text{salt}, K_1||K_2), ek_1||ek_2||c_1||c_2||\text{OtherInput})$$

PRF KDF from SP800-108

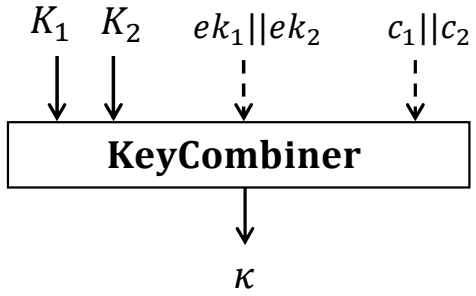


SP 800-227: hybrids

Hybrid KE. How to combine keys? Three approved methods.

3.) Use SP800-133 key combiners.

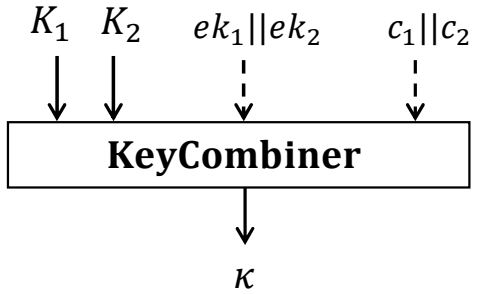
Important: for this method, **all** K_j must be **keys** from **approved** algorithms.



- Concatenation: $\kappa \leftarrow K_1||K_2$
- XORing: $\kappa \leftarrow K_1 \oplus K_2 \oplus \text{OtherInput}$
- HMAC-extraction: $\kappa \leftarrow \underbrace{\mathbf{T}}_{\text{truncation function (see SP800-133)}}\big(\mathbf{HMAC}\ (\text{salt}, K_1||K_2||\text{OtherInput})\big)$

SP 800-227: hybrids

Hybrid KE. How to combine keys?



Much more to say…

- Keys can come from non-KEMs;
- There can be many (i.e., > 2) algorithms involved;
- For methods **1** and **2**, some keys can come from other sources (PSKs, QKD, etc.)
- …

A concern.

- Clearly, there's a lot of choices here…
- (# of KE schemes) × (# of orderings) × (# of combiner options) × (# of KDF options) × …
- How do we deal with this blowup?

Hlep!

Feedback please!

1. **Did we break your stuff?**

   Do any of our **shalls/musts** break your application?

2. **Guardrails for hybrids?**

   Should we explicitly disallow certain combiners? (Even ones that are currently technically allowed by 56C/133/etc.)

3. **Guidance for KEM implicit authentication?**

   E.g., establishing an identity associated to an ek by performing+confirming a KEM key establishment.

4. **Guidance for key confirmation?**

   Does our KC guidance work for your application? Are there other methods you prefer were approved (and if so, why?)

5. **Ephemeral vs static?**

   Guidance (or requirements) regarding the use of ephemeral KEM keypairs?

Questions?

… and anything else you want to tell us.

Specific shalls (Section 4)

RS1 (Section 4.1) KEM implementations **shall** comply with the specific NIST FIPS or SP that concretely specifies the algorithms of the relevant KEM. For example, implementations of ML-KEM **shall** comply with FIPS 203 [3]. *(Note: the CMVP will perform random input-output tests in an attempt to ascertain whether this requirement is satisfied. Ensuring full functional equivalence to the specification via testing is not possible; see also the "**must**" requirement **RM1** below.)*

RS2 (Section 4.1) KEM implementations **shall** comply with the guidance given in FIPS 140-3 [5] and associated implementation guidance.

RS3 (Section 4.1) KEM implementations **shall** use **approved** components with security strengths that are chosen appropriately for each KEM parameter set.

RS4 (Section 4.1) Random bits **shall** be generated using **approved** techniques, as described in the latest revisions of SP 800-90A, SP 800-90B, and SP 800-90C [6–8].

RS5 (Section 4.2) Except for random seeds and data that can be easily computed from public information, all intermediate values used in any given KEM algorithm (i.e., KeyGen, Encaps, and Decaps) **shall** be destroyed before the algorithm terminates.

Specific shalls (Section 5)

RS6 (Section 5.4.1) When a nonce is used by the decapsulator during key confirmation (as specified herein), a nonce with a bit length (at least) equal to the targeted security strength of the KEM key-establishment process **shall** be used (see Appendix A.3).

RS7 (Section 5.4.1) For key confirmation, the MAC algorithm and KC_Key used **shall** have security strengths equal to or greater than the security strength of the KEM and parameter set used.

RS8 (Section 5.4.2) The KC_Key **shall** only be used for key confirmation and destroyed after use.

RS9 (Section 5.5.1) In multi-algorithm key-establishment schemes, shared secrets **shall** be combined via an approved key-combiner, as described in Section 5.5.2.

Specific shalls (Appendices)

RS10 (Appendix A.1) When key confirmation requires the use of a MAC, it **shall** be an approved MAC algorithm (i.e., HMAC, AES-CMAC, or KMAC).

RS11 (Appendix A.1) When a MAC tag is used for key confirmation, an entity **shall** compute the MAC tag on received or derived data using a MAC algorithm with a $MacKey$ that is determined from a shared secret key.

Specific musts

RM1 (Section 4.1). Implementations **must** correctly implement the mathematical functionality of the target KEM. *(Note: the CMVP will perform random input-output tests in an attempt to ascertain whether this requirement is satisfied. Ensuring full functional equivalence to the specification is not possible.)*

RM2 (Section 5.2) In applications of KEMs, a parameter set with application-appropriate security strength **must** be selected (see [9, Section 2.2]).