Advanced FO Concepts: Verifiable Decapsulation, Multi-user Security, and Rejection Modes

NIST Workshop on Guidance for KEMs

Lewis Glabush, Felix Günther, **Kathrin Hövelmanns**,
Mikhail Kudinov, Douglas Stebila

KEMs in the NIST standardization process

Shared approach: PKE from hardness assumption + Fujisaki-Okamoto 'recipe'

Fujisaki-Okamoto (FO): 'generic' encryption-to-key-encapsulation recipe

- 

= moduleLWE encryption, plugged into FO

This talk

- **Security of FO's rejection modes**
- Mitigating faulty implementations (Verifiable Decapsulation)
- Security when FO-based KEMs are used in many interactions

Fujisaki-Okamoto KEMs: initial idea

Goal: Find a public-key method to securely establish symmetric keys K_{sym}.

You may use a public-key encryption scheme and a hash function.

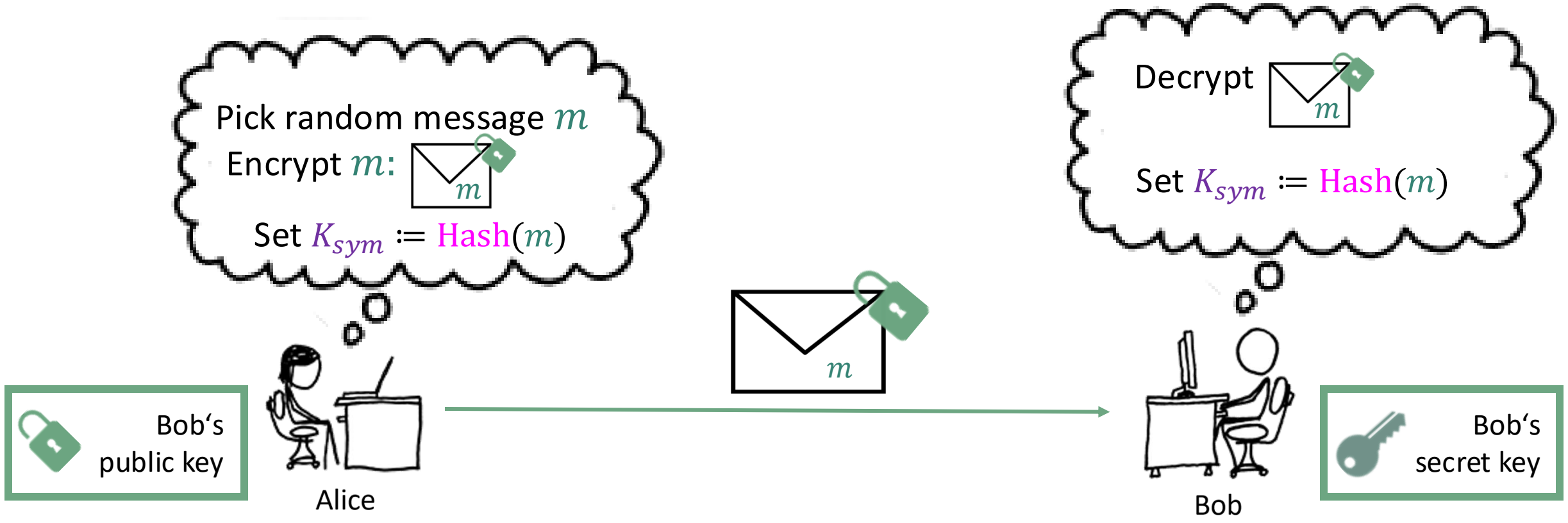


Image source: xkcd.com

Security Target: Chosen-Ciphertext Security

Attacker could **request decapsulation for any ciphertext.**

→ alter how the KEM en-/decapsulates:

Altered decapsulation will

- detect dishonest ciphertexts
- punish those by rejecting to return a meaningful key.

→ attacker can't request <u>useful</u> decapsulations

Fujisaki-Okamoto KEMs

If encryption is probabilistic:

Instead of randomly sampling encryption randomness r,

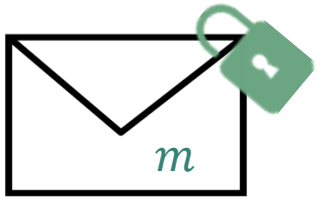
Use $r = \mathrm{Hash}'(m)$

Pick random message m

Encrypt m, **deterministically**

Set $K_{sym} := \mathrm{Hash}(m)$

Alice



Bob

Bob's public key

Bob's secret key

Image source: xkcd.com

Fujisaki-Okamoto KEMs

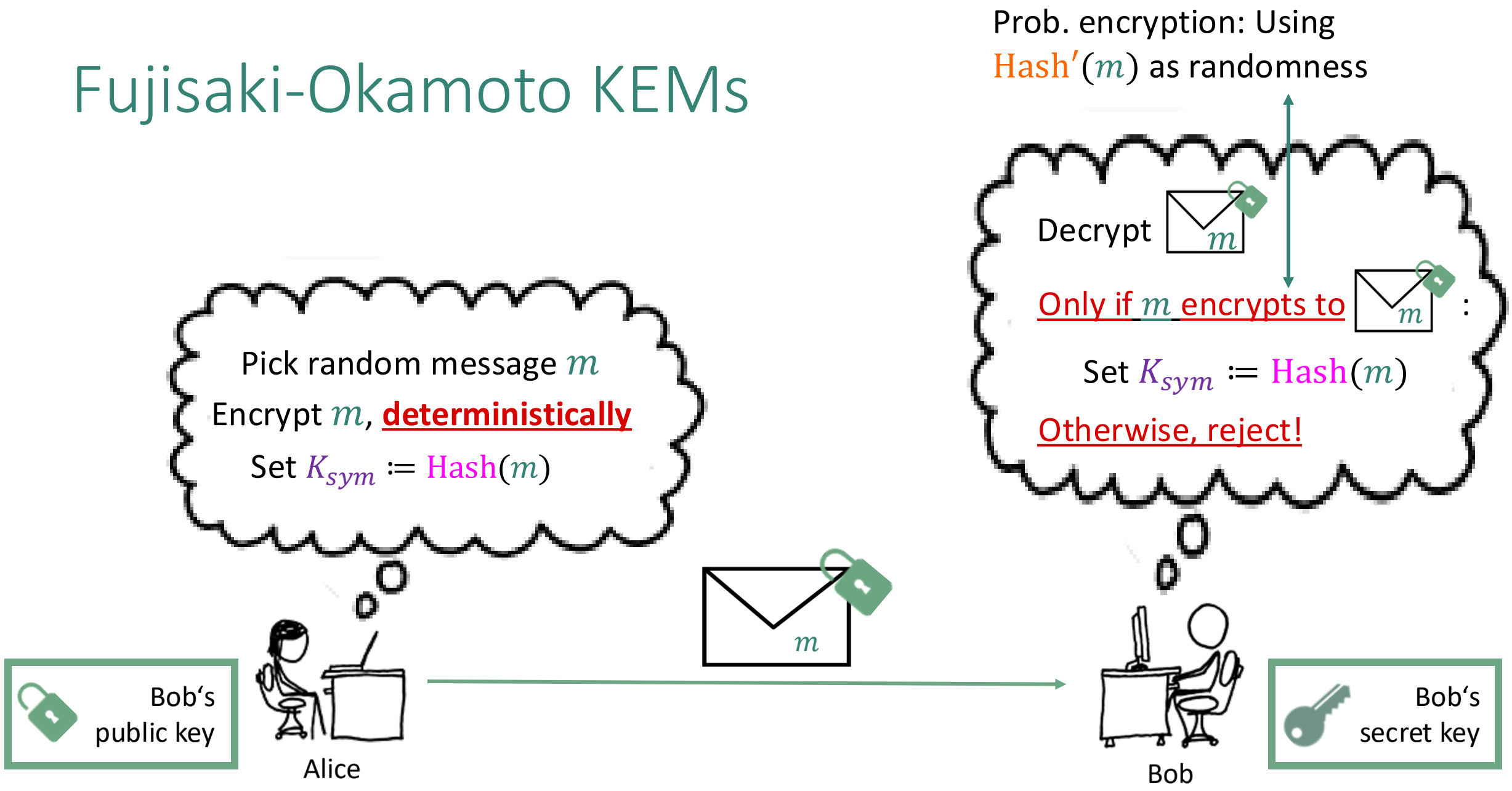
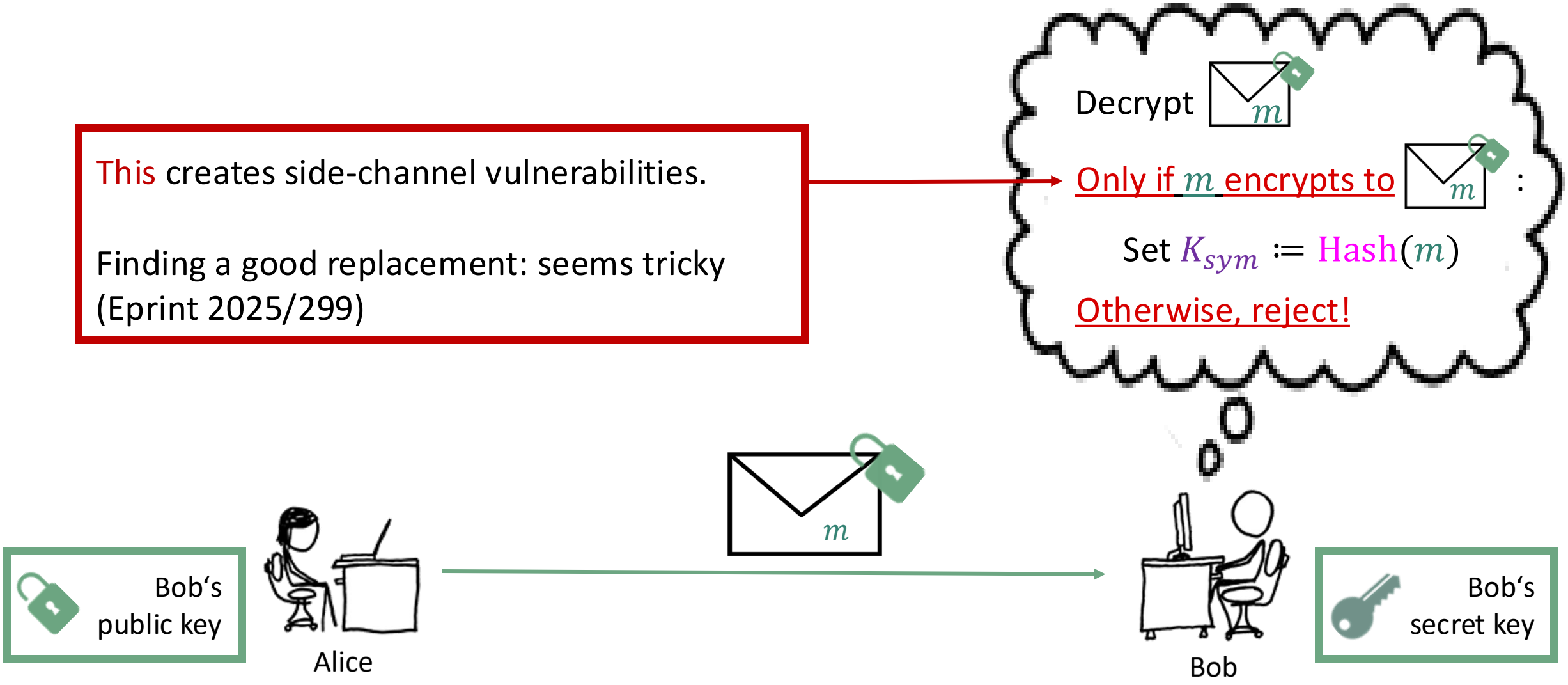


Image source: xkcd.com

Re-encryption check

This creates side-channel vulnerabilities.

Finding a good replacement: seems tricky
(Eprint 2025/299)



[eprint 2025/299]: Hövelmanns, Hülsing, Majenz, Sisinni. (Un)breakable curses - re-encryption in the Fujisaki-Okamoto transform

Implicit vs explicit reject

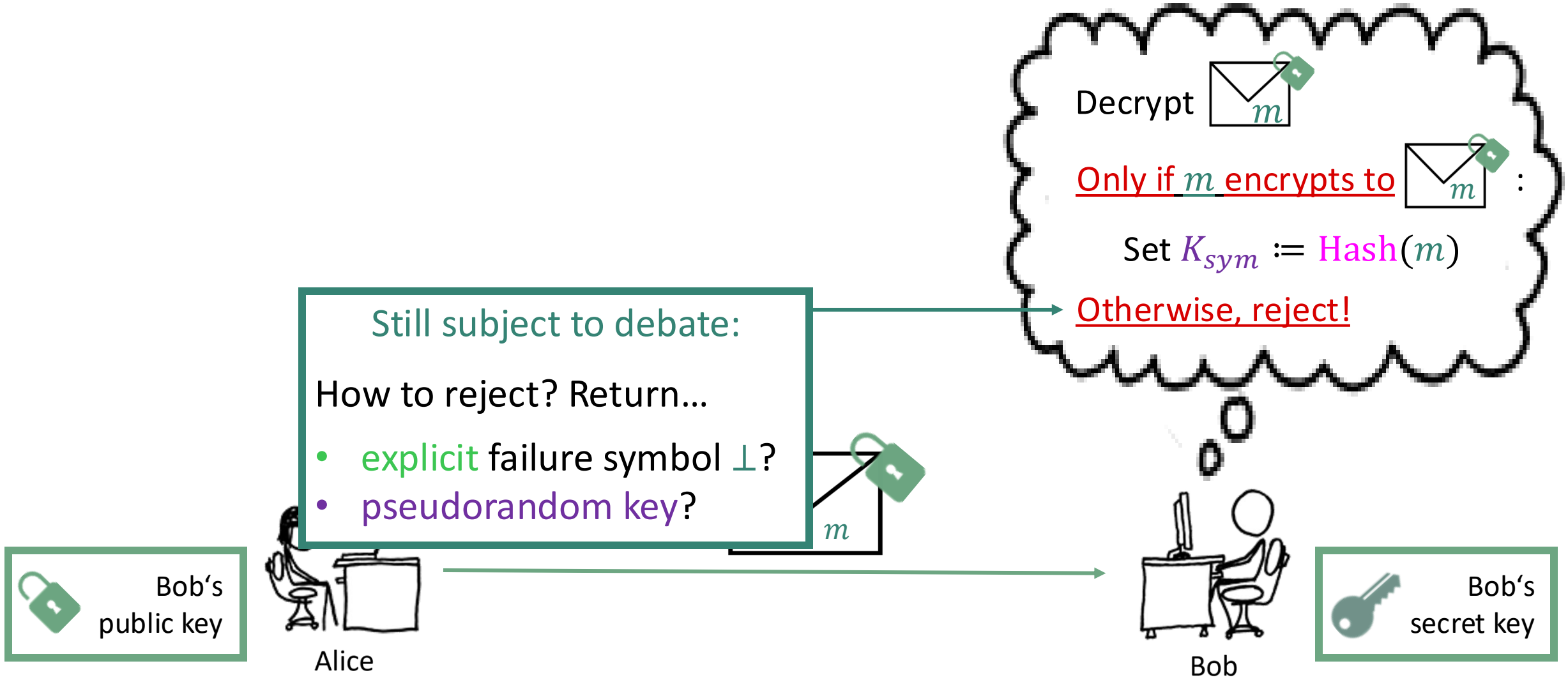


Image source: xkcd.com

Do the rejection modes differ in security?

Intuition: ‘hides rejection branch’

…but does it, in practice?

Implicit: proofs available much earlier*, then tighter

Explicit: additional ‘key confirmation’ hash (until [Zha19])

Still subject to debate:

How to reject? Return…

- explicit failure symbol $\perp$?
- pseudorandom key?

Decrypt m

Only if m encrypts to m :

Set $K_{sym} := \mathrm{Hash}(m)$

Otherwise, reject!

m

Bob‘s public key

Alice

Bob

Bob‘s secret key

Image source: xkcd.com

[Zha19] How to Record Quantum Queries, and Applications to Quantum Indifferentiability

Do the rejection modes differ in security?

1st proof for explicit [DFMS21]: much bigger tightness loss

[HHM22] : bounds closer to implicit rejection variant…

… for **probabilistic** PKE,

with certain correctness properties

Still subject to debate:

How to reject? Return…

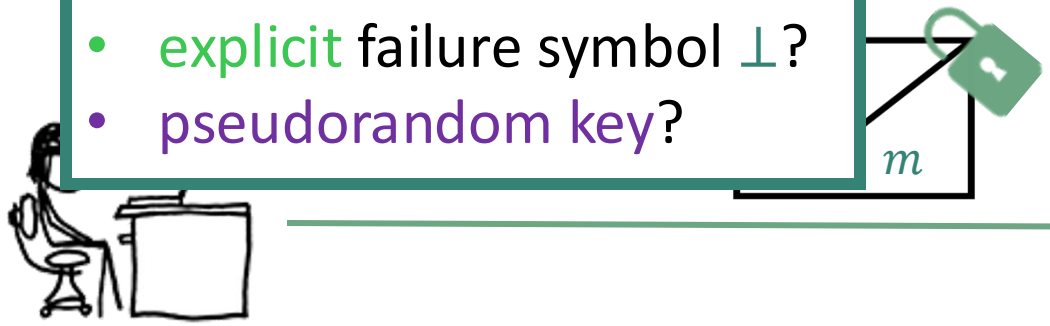
- explicit failure symbol ⊥?
- pseudorandom key?

Decrypt m

Only if m encrypts to m :

Set $K_{sym} := \mathrm{Hash}(m)$

Otherwise, reject!



Bob's public key

Alice

Bob

Bob's secret key

[DFMS21] Online-extractability in the quantum random-oracle model

[HHM22] Failing gracefully: Decryption failures and the Fujisaki-Okamoto transform

Do the rejection modes differ in security?

Most recent result (eprint 2025/062):

Explicit rejection essentially as secure as implicit, when implicit is analyzed in the advanced (extractable) QROM

Still subject to debate:

How to reject? Return…

- explicit failure symbol ⊥?
- pseudorandom key?

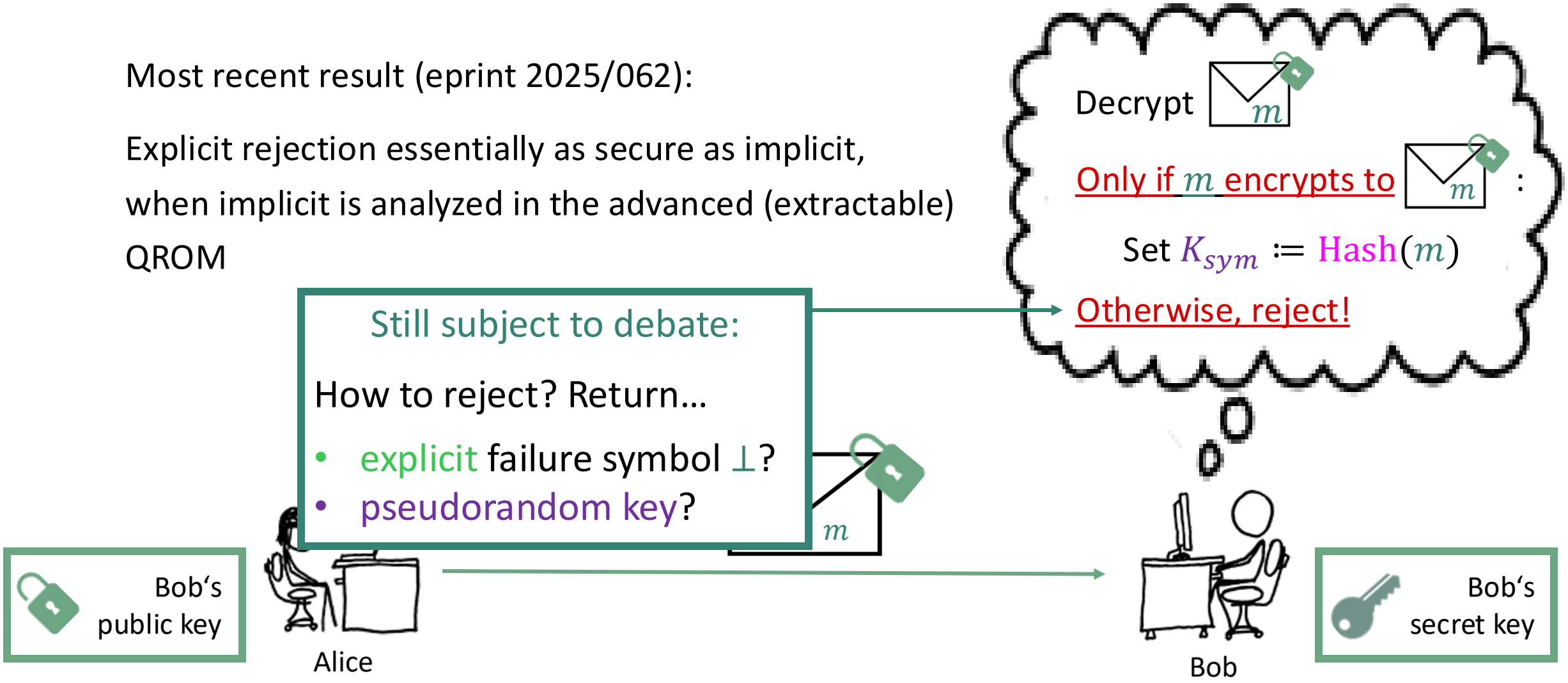


Image source: xkcd.com

[eprint 2025/062]: Hövelmanns, Kudinov. Treating dishonest ciphertexts in post-quantum KEMs

This talk

- Security of FO’s rejection modes
- **Mitigating faulty implementations (Verifiable Decapsulation)**
- Security when FO-based KEMs are used in many interactions

What happens if we skip re-encryption?

We may lose CCA security!

Threat: Eve can perturb ciphertexts, such that

- they decrypt to the same plaintext (same K_{sym}).
- decapsulations leak information about the secret key.

Skipped steps happen in practice! (GoTo Fail, HQC)

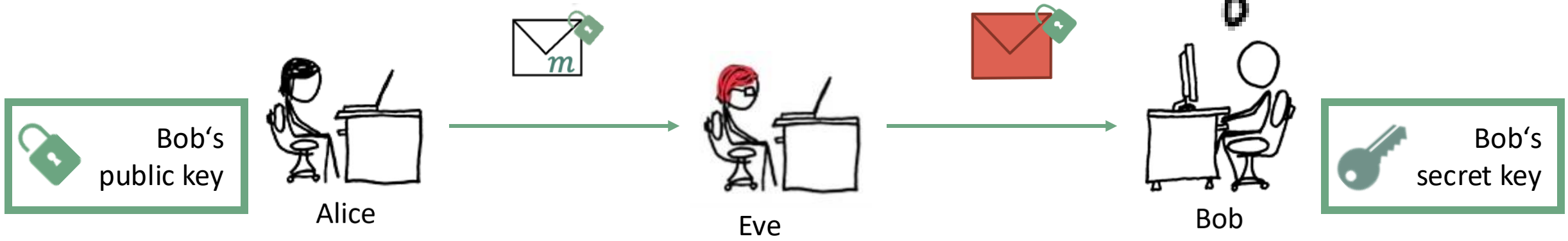


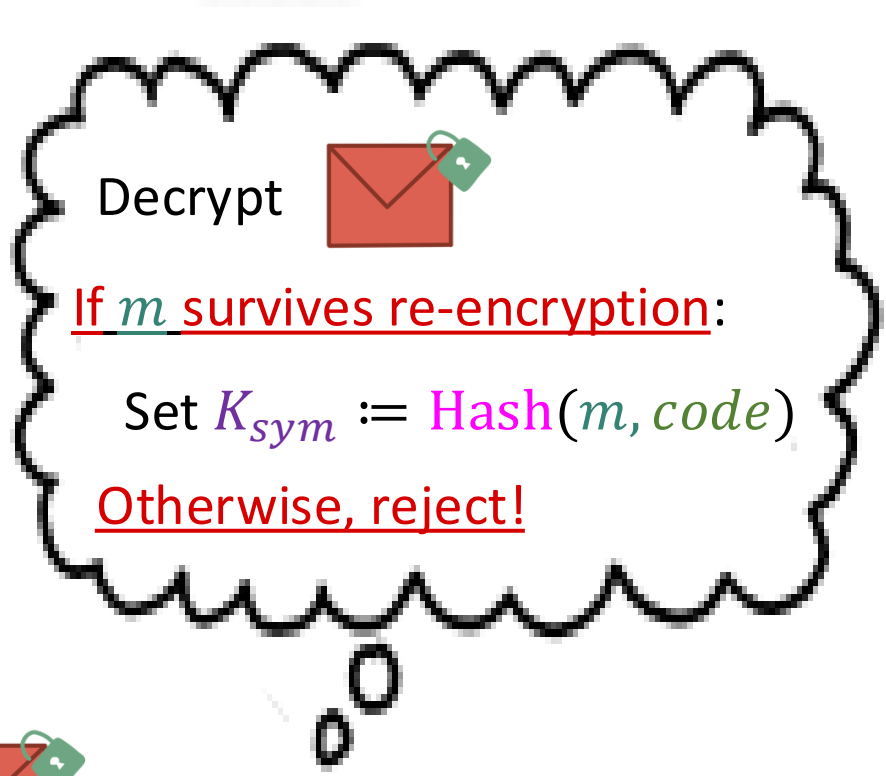
Image source: xkcd.com

Verifiable Decapsulation

Re-encryption is essential for security, but non-essential for functioning correctly.

Solution: alter decapsulation

- tie re-encryption to correct functioning.
- have re-encryption produce a **confirmation code**,
- tie K_{sym} to that code.



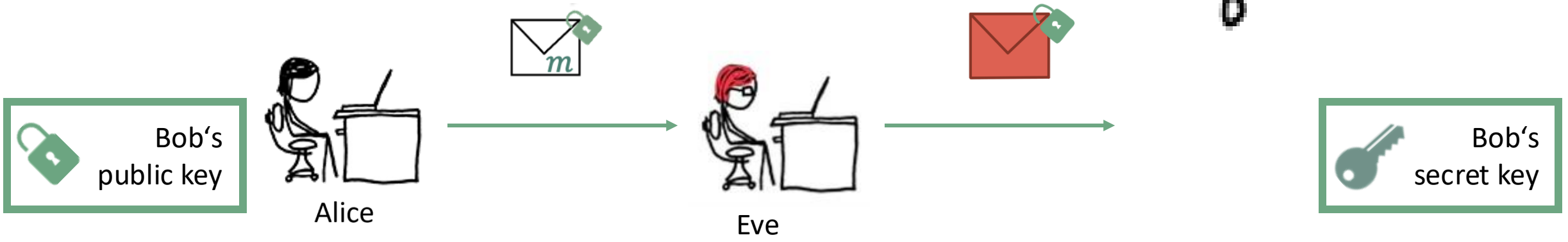


Image source: xkcd.com

Confirmation Code Requirements

Re-encryption now produces a **confirmation code** $code$, used when creating $K_{sym} := \mathrm{Hash}(m, code)$.

$code$ should be

- hard to predict without re-encrypting (security)
- small (low overhead).

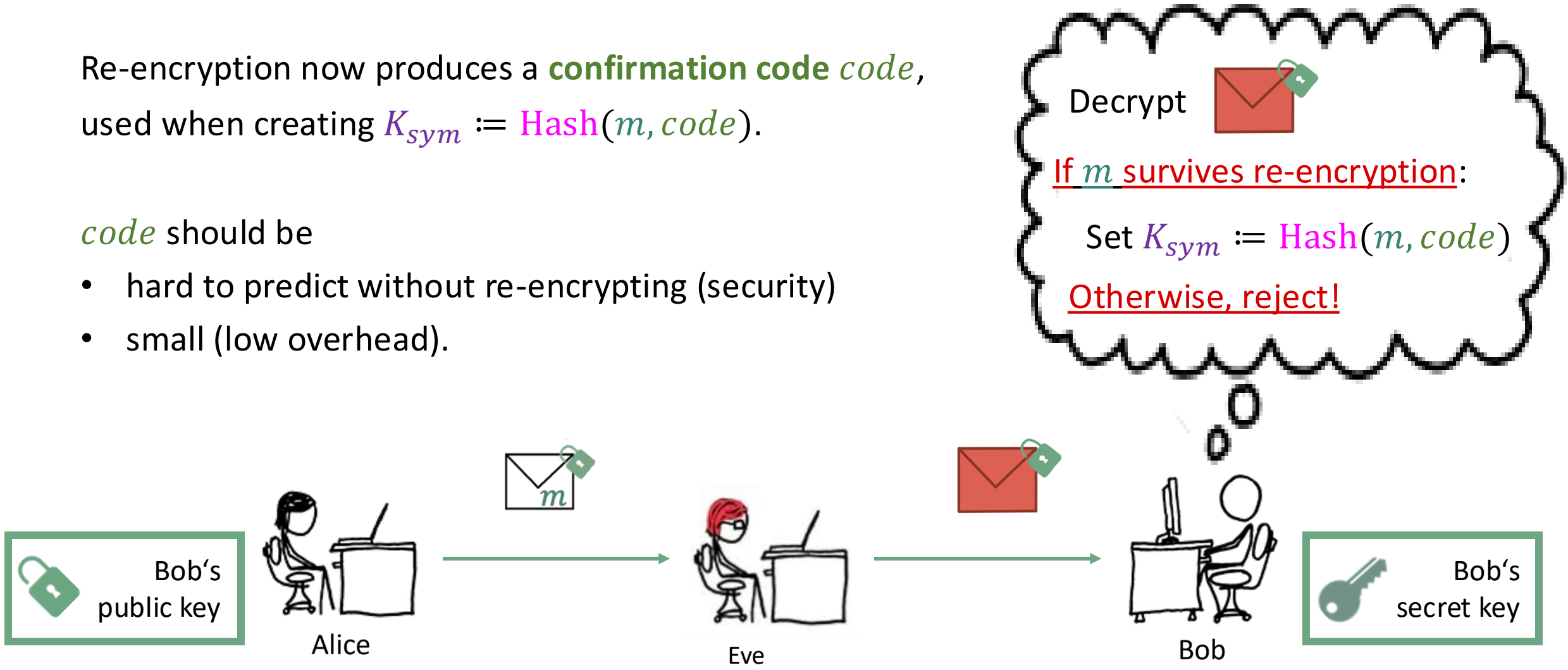


Image source: xkcd.com

Concrete Codes

ML-KEM/Kyber: $code$ derived from error vectors.
Less than 3% slowdown.

HQC: performs ciphertext truncation, $code$ derived from truncated bits.
Less than 0.25% slowdown.

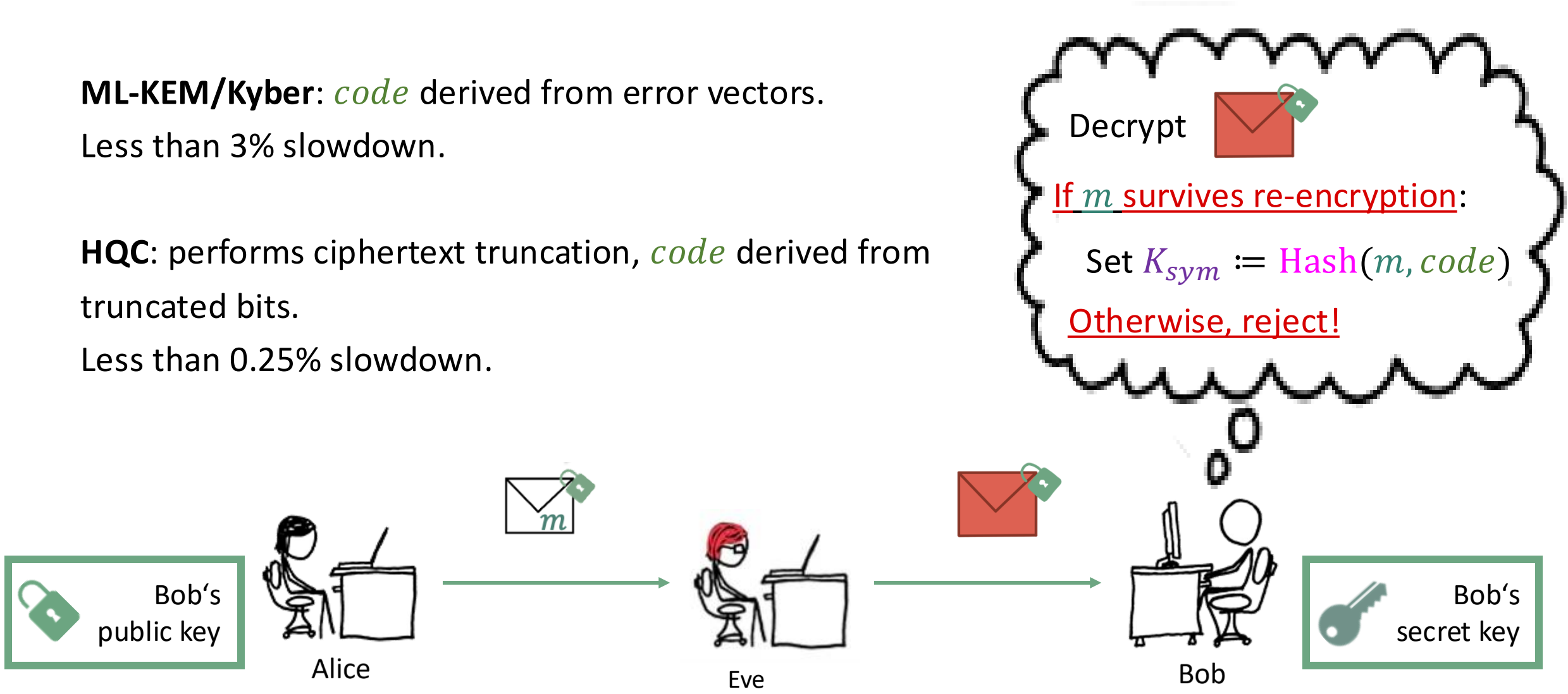


Image source: xkcd.com

This talk

- Security of FO’s rejection modes
- Mitigating faulty implementations (Verifiable Decapsulation)
- **Security when FO-based KEMs are used in many interactions**

Multi-Challenge Security

Limitation of IND-CCA: many users will use this KEM, sending many messages

→ we actually need multi-user, multi-ciphertext INDCCA

Multi-user security [DHKLS21]: Use domain separation to bind Bob's identity (a prefix $pref$ of the public key pk) to

- how we define validity of a ciphertext:

  use $\mathrm{Hash}'(\boldsymbol{pref}, m)$ as encryption randomness

- how the symmetric key is computed:

$$K_{sym} := \mathrm{Hash}(\boldsymbol{pref}, m)$$

→ hard for attacker to exploit information related to Bob to attack Carol.

[Duman Hövelmanns Kiltz Lyubashevsky Seiler 21]: Faster lattice-based KEMs via a generic Fujisaki-Okamoto transform using prefix hashing.

Multi-Ciphertext Security

Limitation of IND-CCA: many users will use this KEM, sending many messages

Multi-ciphertext security: The FO transform has deterministic encryption, which can lead to small ciphertext spaces.

Problem: For KEM's with "small" message space, like FrodoKEM640 and HQC128, the birthday effect leads to collision attacks.

Solution: We mitigate this by adding a *salt* to the FO transform, leading to much tighter bounds.

Change how the symmetric key is computed:

$$K_{sym} := \mathrm{Hash}(\boldsymbol{pref}, m||salt)$$

Impact of Salting (Intuitively)

Limitation of IND-CCA: many users will use this KEM, sending many messages

$$K_{sym} := \mathrm{Hash}(\boldsymbol{pref}, m||salt)$$

KEM to PKE multi-challenge security reduction:

$$\mathrm{Adv}_{\mathrm{KEM}}(A) \leq 2\mathrm{Adv}_{\mathrm{PKE}}(B) + \text{“collisions”} + \text{“correctness errors”}.$$

Salting reduced the collision probability by a factor of about 2^{64}.

Multi-Target Security

Limitation of IND-CCA: many users will use this KEM, sending many messages

→ we actually need multi-user, multi challenge INDCCA

Multi-user security: Use domain separation to bind Bob's identity (a prefix $pref$ of the public key pk) to

- how we define validity of a ciphertext:

  use $\mathrm{Hash}'(\boldsymbol{pref}, m)$ as encryption randomness

- how the symmetric key is computed:

$$K_{sym} := \mathrm{Hash}(\boldsymbol{pref}, m)$$

→ hard for attacker to exploit information related to Bob to attack Carol.

[Duman Hövelmanns Kiltz Lyubashevsky Seiler 21]: Faster lattice-based KEMs via a generic Fujisaki-Okamoto transform using prefix hashing.

Thanks for listening!

Result 1: Rejection modes are almost-equivalent

- increases trust in explicit rejection

  + when using FO-based KEMs in bigger protocols

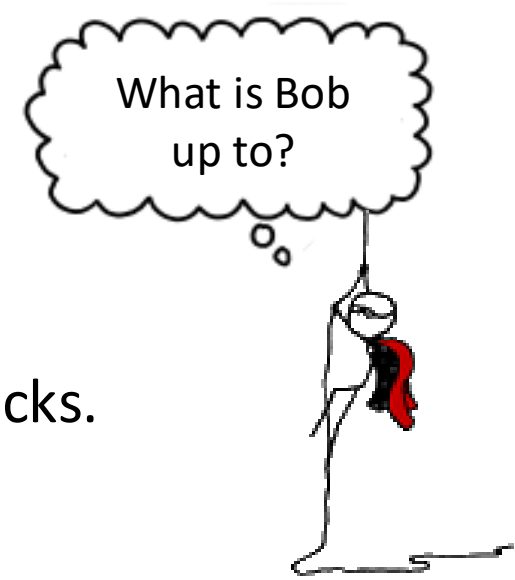
Result 2: Verifiable Decapsulation

- mitigates vulnerabilities stemming from lazy/faulty implementations

Result 3: “Salted” FO

- Enables multi-target security for encryption schemes with “small” message space

Security of KEMs: IND-CCA



IND-CCA security: **IND**istinguishability under **C**hosen-**C**iphertext **A**ttacks.

Adversary plays one of two games, needs to guess which one:

Left game	**Right game**
Adversary gets public key Adversary gets ciphertext c that 'encrypts' a symmetric key K_{sym}, together with	
K_{sym} belonging to c	Uniformly random K_{sym}

Chosen-ciphertext: Adversary can request decryptions for any ciphertext it chooses (except 'challenge' ciphertext c)

Image source: xkcd.com

Proof technique: extractable QROM

Idea: ROM-like reduction via preimage extraction

QROM $O: X \to Y$ via compressed oracle (Zha19)

+ interface $\mathrm{Extract}_f$ for $f: X \times Y \to T$:

$\mathrm{Extract}_f(\mathrm{t})$:

Collapse oracle database such that

- for one x, $f(x, y) = t$ for all y that are in the database superposition for x

Return x

In FO proof:

$O = \mathrm{Hash}_{\mathrm{rand}}$: $M \to R$

$f =$ Encrypt: $M \times R \to C$

$\mathrm{Extract}_f(\mathrm{c})$ = 'preimage' m

'Surprising' $\triangleq$ PKE spreadness

$\mathrm{Extract}_f$ commutes nicely with O-operations for sufficiently surprising f.

Compressed oracle (Zha19)

- Oracle database initalised to $D := \bigotimes_{x \in query\ domain} |x, \perp>_{D_x}$
- Process queries $|x, y>$ by applying
  - F_{D_x} to output register of D_x

$$F_{D_x}|\psi> := \begin{cases} uniform\ superposition, & |\psi> = \perp \\ \perp, & |\psi> = uniform\ superposition \\ |\psi>, & |\psi> orthogonal\ to\ \perp, uniform \end{cases}$$

  - $\mathrm{CNOT}^{\otimes}_{D_x:Y}$ to D_x, query output register *Y*
  - F_{D_x} to output register of D_x