Explicitly rejecting Fujisaki-Okamoto and worst-case correctness

pqCrypto 2024

Kathrin Hövelmanns

June 12th, 2024, Oxford

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Motivation: KEMs + the NIST process

Key Encapsulation Mechanisms are

- one of NIST's 2 pq standardization aims
- public-key methods to securely establish a symmetric key K_{sym}.

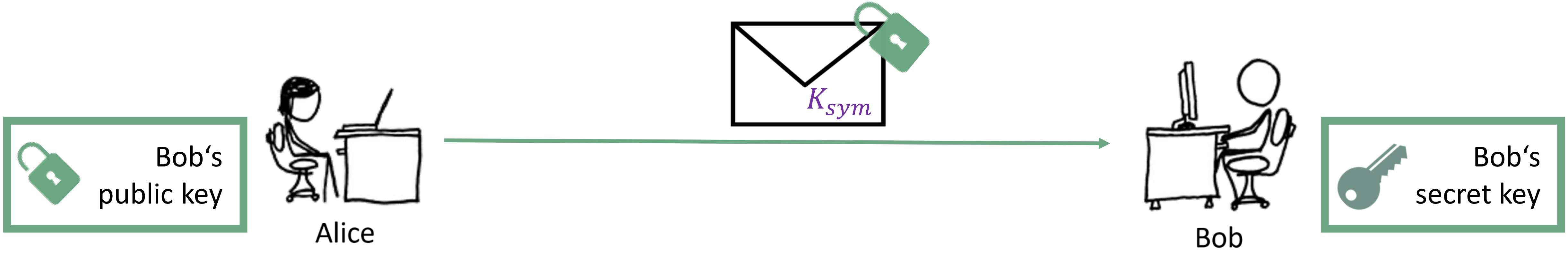
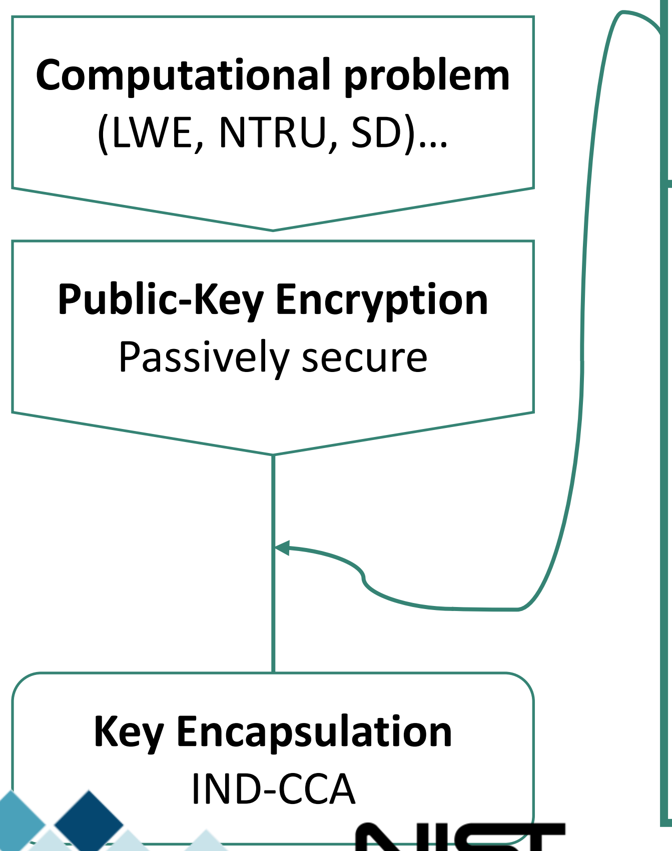


Image source: xkcd.com

Motivation: KEMs + the NIST process



Fujisaki-Okamoto: ‘generic’ PKE-to-Key-Encapsulation recipe, e.g.

KYBER = FO, applied to moduleLWE encryption

HHK17: proofs deal with

☑ occasional decryption failures (lattices, codes)

☑ quantum attacks (quantum ROM)

but…

QROM: proof only for somewhat-unnatural variant,

suboptimal bounds

National Institute of Standards and Technology
U.S. Department of Commerce

FO KEMs: initial idea

Goal: Establish a symmetric key K_{sym}, using a PKE scheme and a hash function.

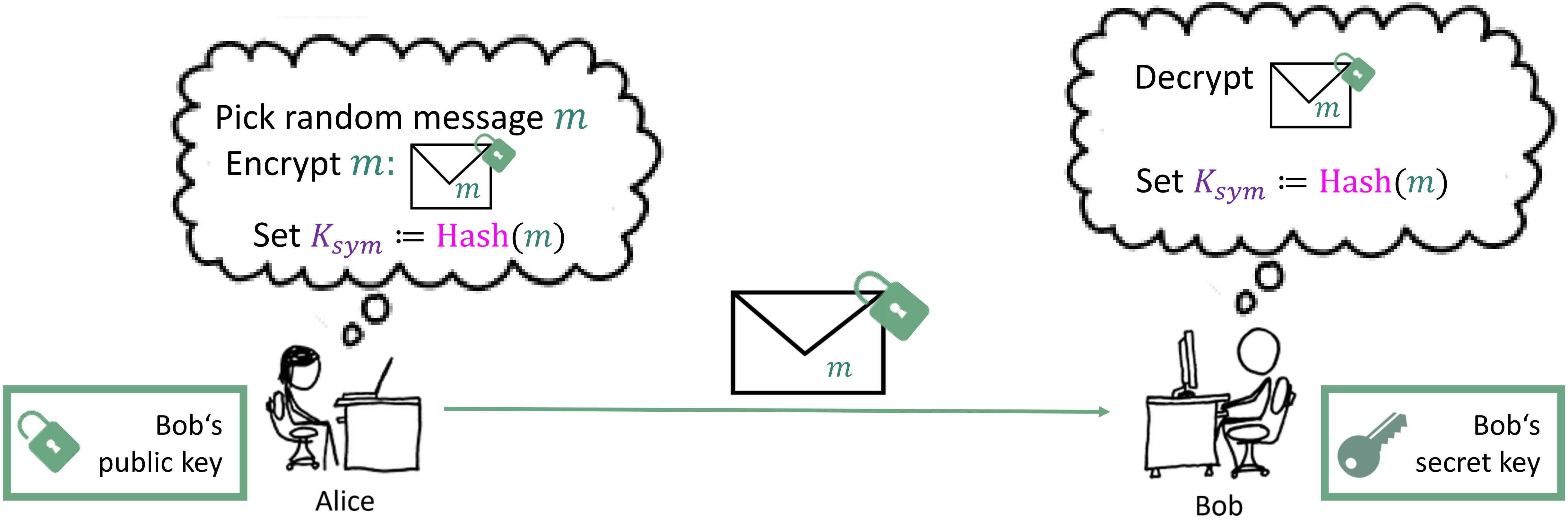


Image source: xkcd.com

FO KEMs: IND-CCA security

Goal:

Security, even if attackers can **request decapsulations**

How?

Alter decapsulation: Prevent that such requests are useful

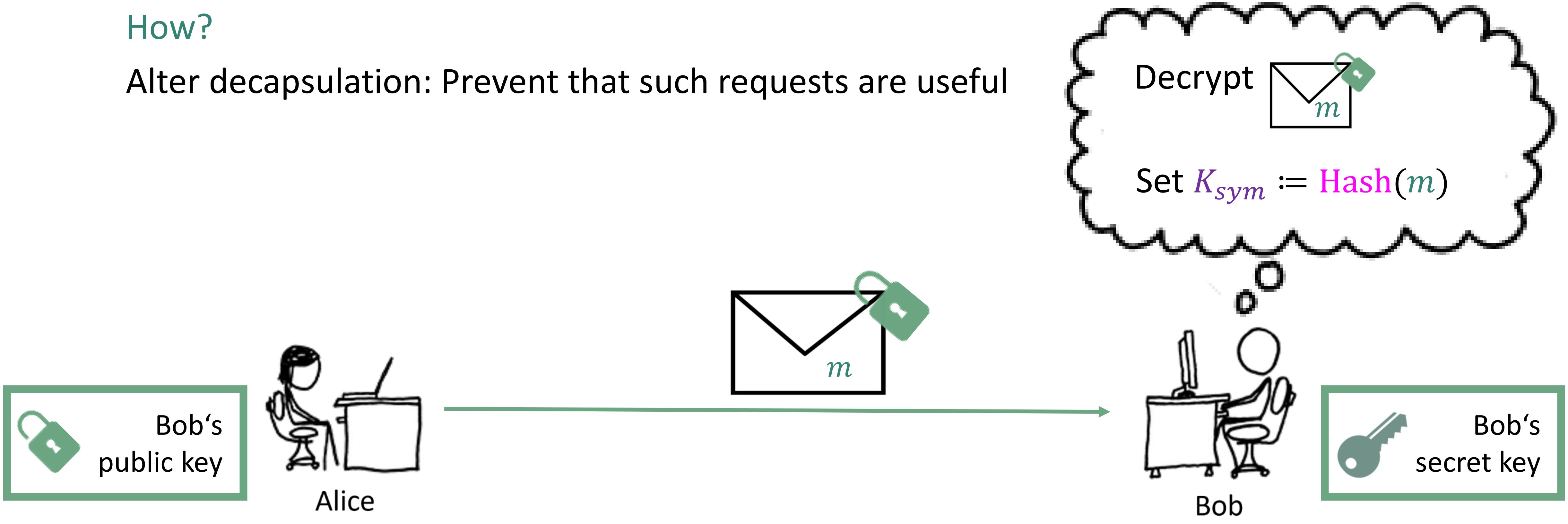


Image source: xkcd.com

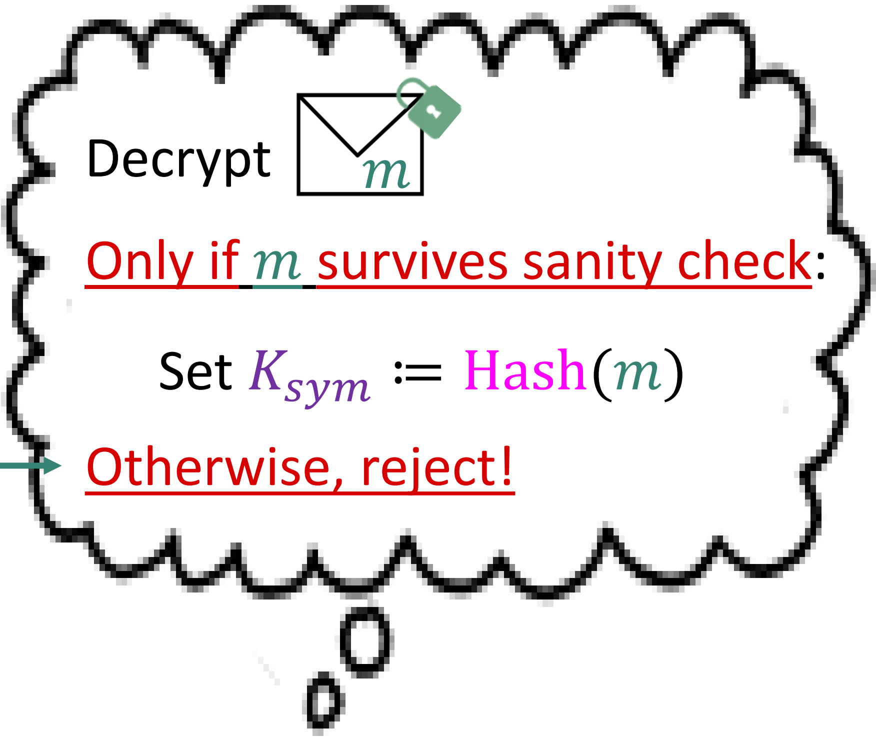
FO KEMs: IND-CCA security

Goal:

Security, even if attackers can **request decapsulations**

How?

Alter decapsulation: Prevent that such requests are useful



Still subject to debate:

How to reject? Return…

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

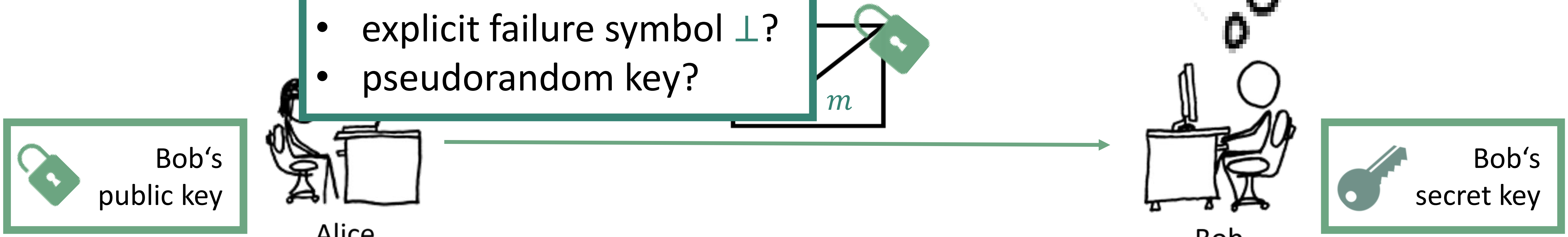


Image source: xkcd.com

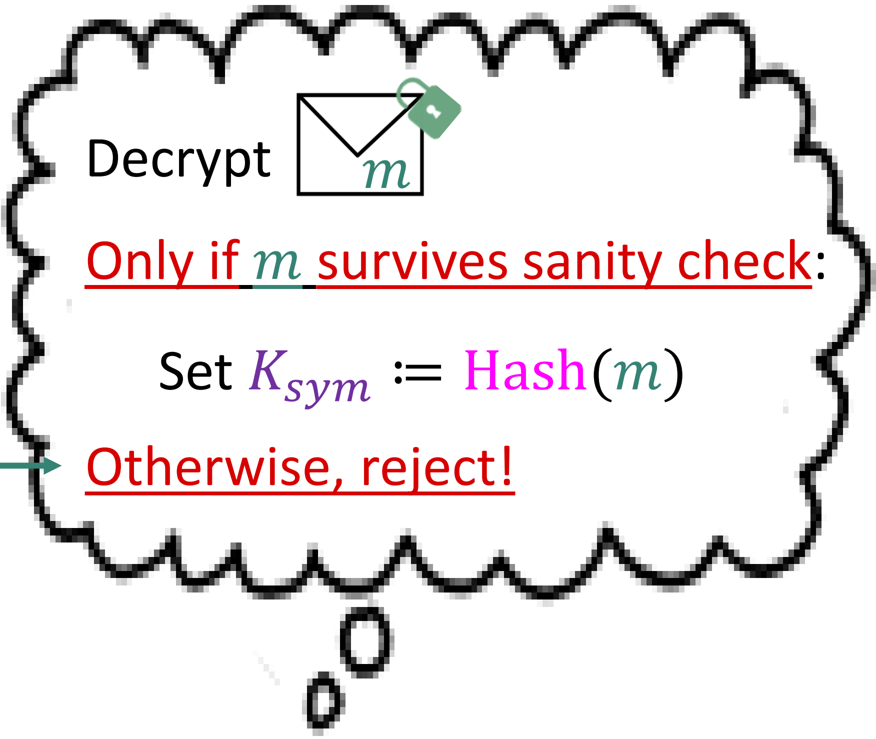
Implicit vs explicit reject

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?

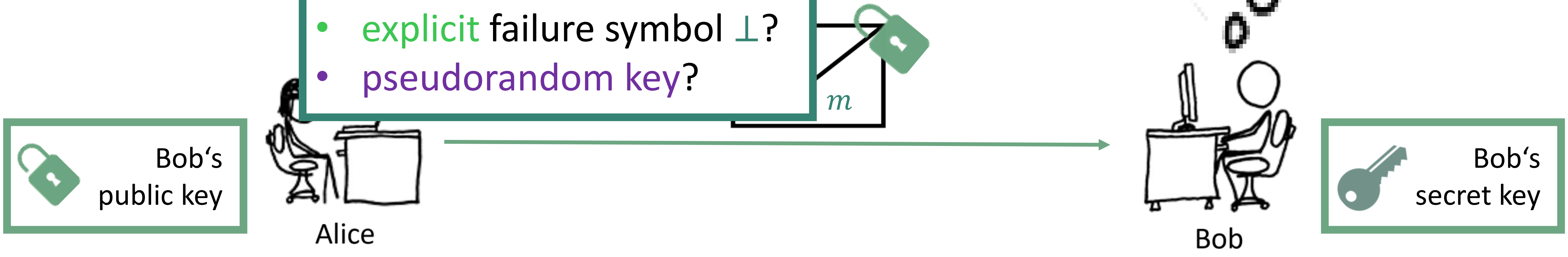


Image source: xkcd.com

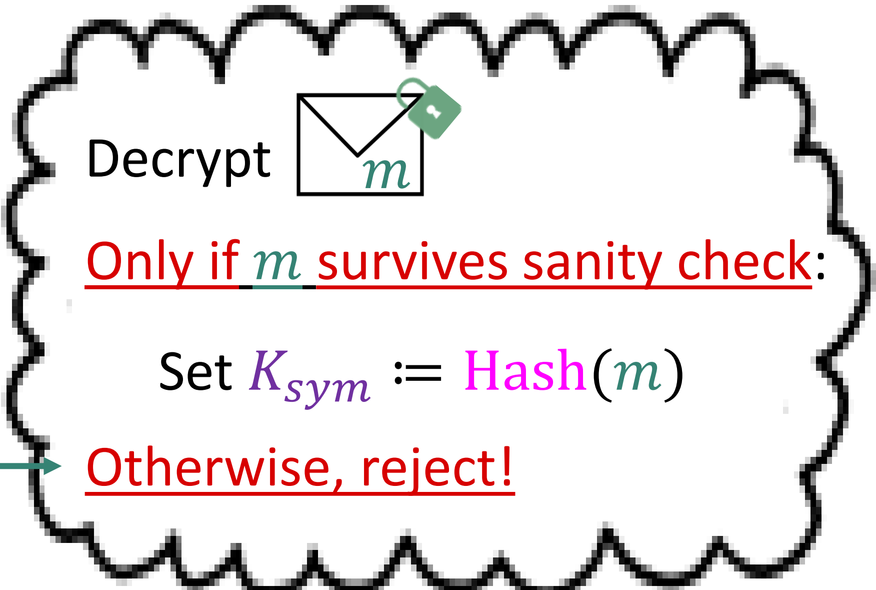
* [SXY18, JZ+18, BHH+19, HKSU20, KSS+20]

Explicit reject in the QROM after [Zha19]

First proof [DFMS21]: much bigger tightness loss

[HHM22] got bounds closer to implicit-rejection …

… for **probabilistic** PKE **with certain correctness properties**



Still subject to debate:

How to reject? Return…

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

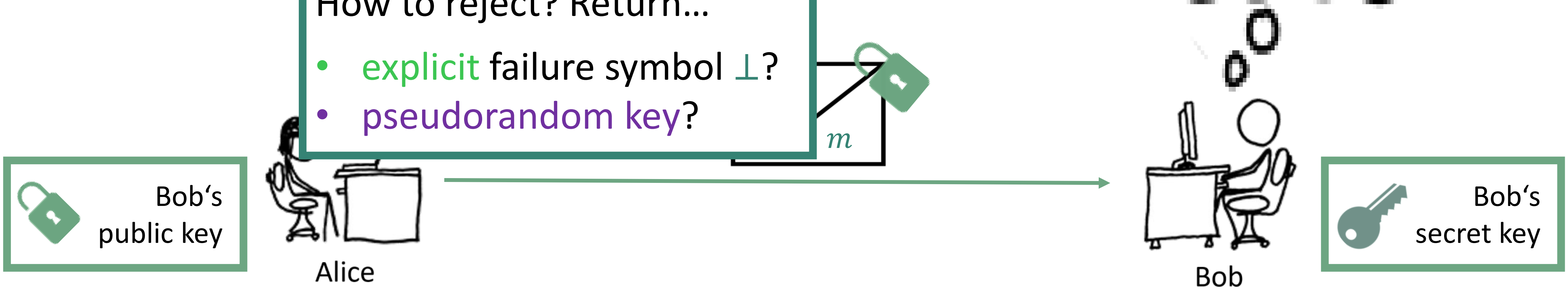


Image source: xkcd.com

Imperfect correctness

With some probability,

$$Decrypt(Encrypt(m)) \neq m$$

leakage on secret key

[DGJ+19, BS20, DRV20, FKK+22]

HHK17: Upper-bound per-m failure probability by δ

☺ hard to even find failing ciphertexts

☹ bounds so far only heuristic

Explicit reject and imperfect correctness

[HHM22] bound for explicitly rejecting FO ($\mathrm{FO}^{\perp}$), applied to probabilistic scheme PKE:

$$\text{IND-CCA}\,(\mathrm{FO}^{\perp}) \lessapprox \text{IND-CPA}\,(\mathrm{FO}^{\perp}) + \text{Failure–CCA}\left(\mathrm{PKE}^{\mathrm{derand}}\right) + \epsilon_{\gamma}$$

$$\epsilon_{\gamma} \approx \frac{q_D \cdot q}{\sqrt{2^{\gamma}}}$$

γ: PKE spreadness (‘entropy’)

q: # queries to ROs

q_D: # decryption requests (NIST: 2^{64})

Explicit reject and imperfect correctness

[HHM22] bound for explicitly rejecting FO ($FO^{\perp}$), applied to probabilistic scheme PKE:

$$\text{IND-CCA}\ (FO^{\perp}) \lesssim \text{IND-CPA}\ (FO^{\perp}) + \text{Failure–CCA}\left(\mathrm{PKE}^{\mathrm{derand}}\right) + \epsilon_{\gamma}$$

$\mathrm{FAILURE-CCA}\ (\mathrm{PKE}^{\mathrm{derand}})$ in extractable QROM

↳ $\mathrm{NONGENFAIL}\ (\mathrm{PKE}) + \mathrm{GENFAIL}\ (\mathrm{PKE}^{\mathrm{derand}})$

☺ **more fine-grained bounds**

☹ **more work for scheme designers**

Q: Can we replace Failure–CCA with the δ **–heuristic**?

Our result

Bound for explicitly rejecting FO ($\mathrm{FO}^{\perp}$), applied to probabilistic scheme PKE:

$$\text{IND-CCA}\,(\mathrm{FO}^{\perp}) \lessapprox \text{IND-CPA}\,(\mathrm{FO}^{\perp}) + \text{Failure–CCA}\left(\mathrm{PKE}^{\mathrm{derand}}\right) + \epsilon_{\gamma}$$

$$\mathrm{FAILURE} - \mathrm{CCA}\,(\mathrm{PKE}^{\mathrm{derand}}) \lessapprox q^2 \cdot \delta$$

q: # queries to ROs

δ: Upper-bound on per-m failure probability as in [HHK17]

☺ Best of both worlds: Proof for explicit rejection now works for δ –heuristic!

Proof overview

Goal: Failure−CCA $\left(\mathrm{PKE}^{\mathrm{derand}}\right) \lessapprox q^2 \cdot \delta$

Step 1: $\equiv$ chance at success for following task:

- **Task**: Find failing message m:

  m s. th. $Decrypt\left(Encrypt(m)\right) \neq m$

- **Having access to**
  - public and secret key,
  - random oracle used to generate the encryption randomness
  - additional extraction interface $\mathrm{Extract(c)}$ = 'preimage' *m* for c

Intuition: chance at success $\lessapprox q^2 \cdot \delta$ for attackers **without** $\mathrm{Extract}$ **interface**

→ **Step 2**: Show: availability of $\mathrm{Extract}$ has only mild effect on chance at success

Proof overview – step 2

- **Task**: Find failing message m:

  m s. th. $Decrypt\left(Encrypt(m)\right) \neq m$

- **Having access to**
  - public and secret key,
  - random oracle used to generate the encryption randomness
  - additional extraction interface $\mathrm{Extract(c)}$ = 'preimage' *m* for c

Lemma: $\sqrt{\Pr\,[Win]} \lessapprox \sum_{i=1}^{q+1} \max_{m,i} \sqrt{p_{FIND}(m,i)}$

Prob. that i–th oracle query triggers decryption error

Then bound: $\sum_{i=1}^{q+1} \max_{m,i} \sqrt{p_{FIND}(m,i)} \;\leq (q+1) \cdot \sqrt{\delta}$

Conclusion

Thanks for listening!

Eprint: 2023/1811

New bound for $\mathrm{FO}^{\perp}$ for schemes with sufficient entropy:

$$\text{IND-CCA}\ (\mathrm{FO}^{\perp}) \lessapprox \text{IND-CPA}\ (\mathrm{FO}^{\perp}) + q^2 \cdot \delta + \epsilon_\gamma$$

q:	# queries to RO
δ:	Upper-bound on per-m failure probability as in [HHK17]
ϵ_γ:	PKE spreadness (‘entropy’) term

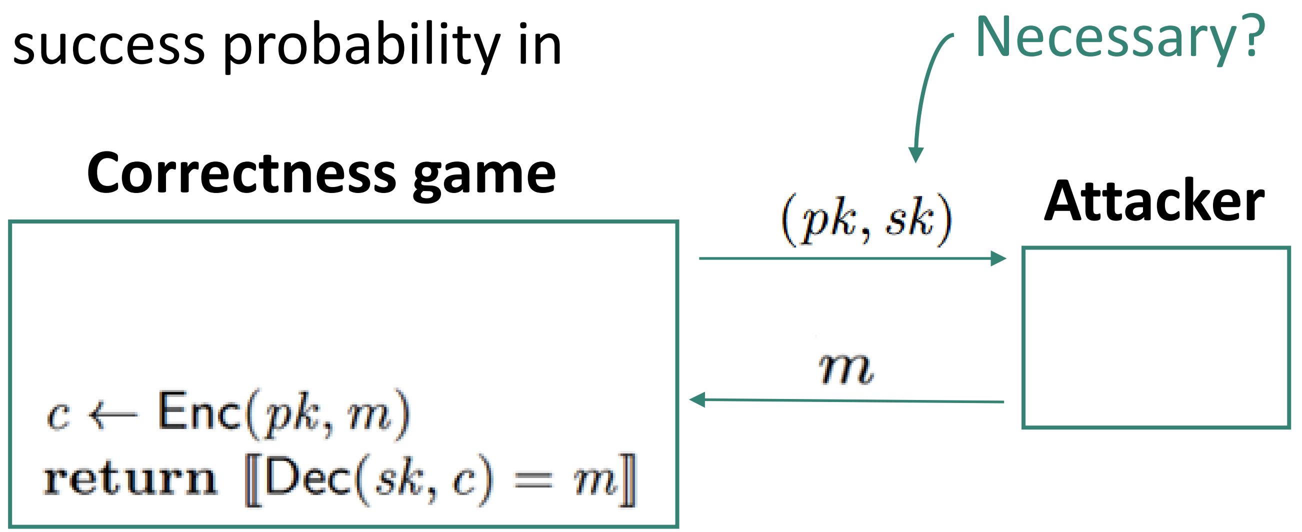
QROM tools: Extending compressed oracles by Extract

- furthers almost-classical reasoning ☺
- without disturbing bounds for oracle search problems

(eg preimages, collisions, predicate fulfillers…)

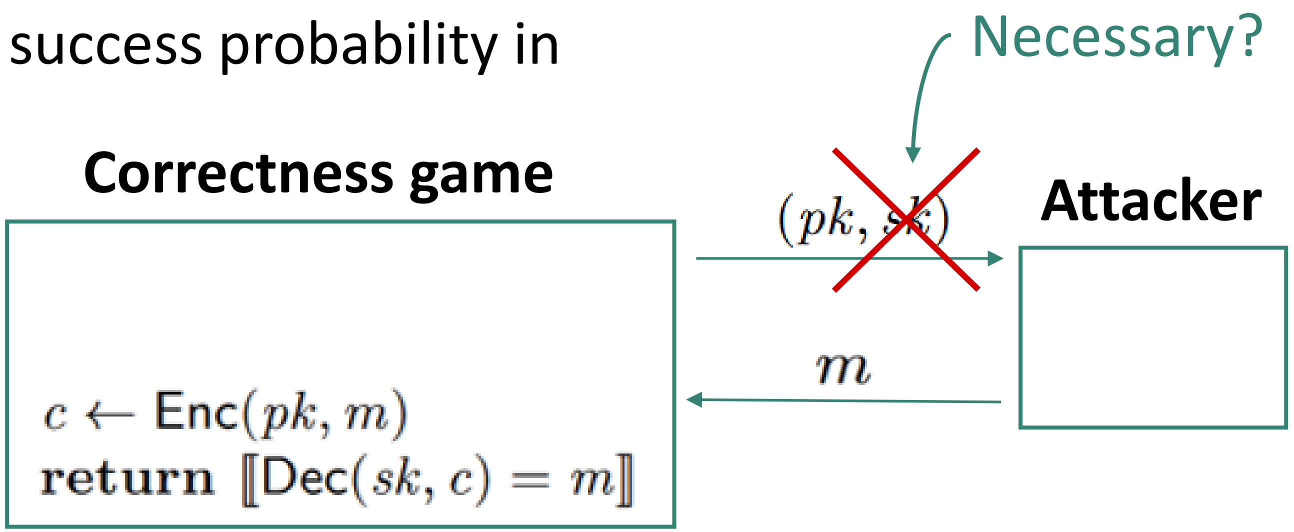
Bonus: δ- estimations vs security proofs

$\delta \triangleq$ success probability in



Bonus: δ- estimations vs security proofs

$\delta \triangleq$ success probability in



⚡observed by Manuel Barbosa while formally verifying Kyber

δ-estimator scripts:

estimate $\triangleq$ success probability in game **without sk**

Applicability issue

Concrete δ – estimations ⚡ security proofs

Bonus: Key indistinguishability + OWTH

Goal: Establish a symmetric key K using a PKE scheme and a hash function

Pick random message m

Encrypt m: [envelope m with lock]

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

IND-CPA security of FO

Random Oracle reasoning:

A cannot do this without querying Hash on m

-> A broke [envelope m with lock]

‘real’ / ’random’

IND-CPA for KEMs:

Seeing [envelope m with lock], A must tell K_{sym} apart from random.

Bob‘s public key

Alice

Bob

Bob‘s secret key

Image source: xkcd.com

Bonus: Key indistinguishability + OWTH

Goal: Establish a symmetric key K using a PKE scheme and a hash function

Pick random message m

Encrypt m:

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

Alice

Bob's public key

Image source: xkcd.com

QROM IND-CPA security of FO

IND-CPA for KEMs:

Seeing [envelope], ***A*** must tell K_{sym} apart from random.

Quantum ROM reasoning:

A cannot do this without querying Hash on m

-> ***A*** broke [envelope]

Kyber etc.: Oneway-to-Hiding (OWTH) [Unruh 14]

Advantage(***A***) $\lessapprox q\sqrt{\epsilon}$

q = # queries to Hash

ϵ = Advantage against [envelope]

Bound improvements:

Advantage(***A***) $\lessapprox \begin{cases} \sqrt{\epsilon} & \text{[BH+ 19]} \\ q\epsilon & \text{[KS+ 20]} \end{cases}$

(optimal? still tbd)

Proof technique: Extractable QROM [DFMS22]

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 O's database such that

- for one x, $f(x, y) = t$ for all y in x's database superposition

Return x

FO proof:

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

$f = \mathrm{Encrypt}: M \times R \to C$

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

'Surprising' ≜ PKE spreadness

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