Cryptography and Quantum Key Distribution

2022-10-03: IOTALENTUM WORKSHOP TTW2 – CYBERSECURITY AND APPLICATIONS

S. R. Verschoor

Department of Electrical Engineering

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Outline

- Cryptography
  - Basics
  - Post-Quantum Cryptography (PQC)
- Quantum Key Distribution (QKD)
  - QKD Network
  - TU/e testbed

Outline

- Cryptography
  - Basics
  - Post-Quantum Cryptography (PQC)
- Quantum Key Distribution (QKD)
  - QKD Network
  - TU/e testbed

Cryptography – the basics

- Alice and Bob want to communicate
  - Mallory is actively interfering with them
    - (in some weaker models Eve is only passively eavesdropping)

- Kerckhoff's principle
  - aka Shannon's Maxim: "the enemy knows the system"
  - but Mallory does not know the keys

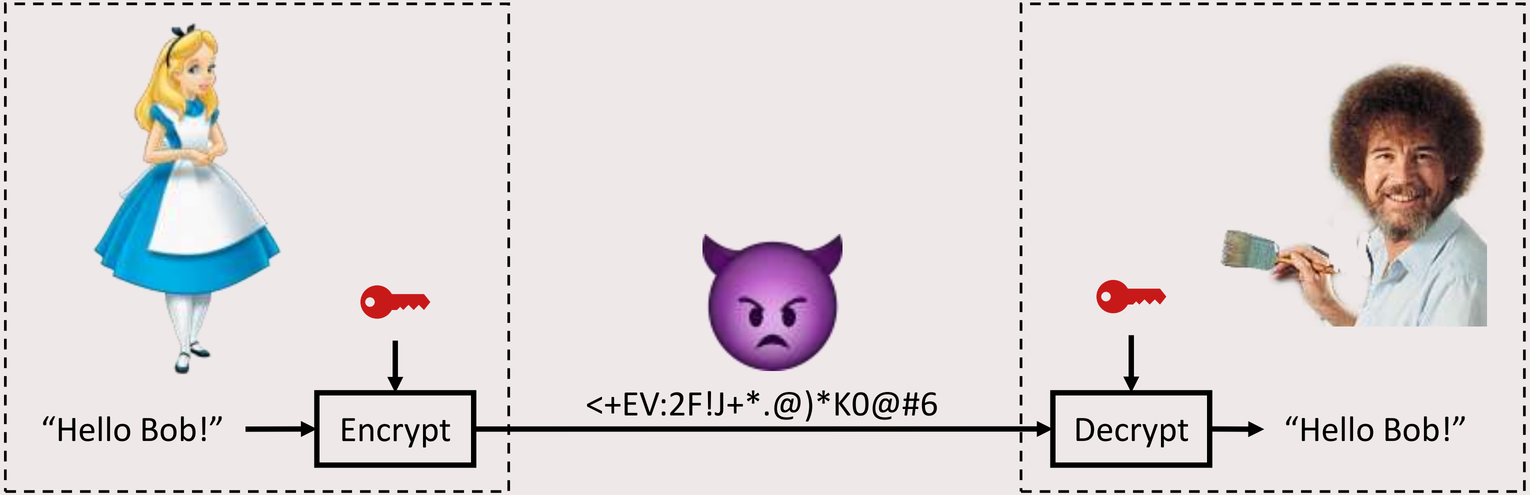
- Mallory carries the messages (Dolev-Yao model)
  - she can inspect, change, re-order, replay, drop, inject any message
  - may (sometimes) compromise some participants

Confidentiality

- Alice and Bob want their message to remain secret

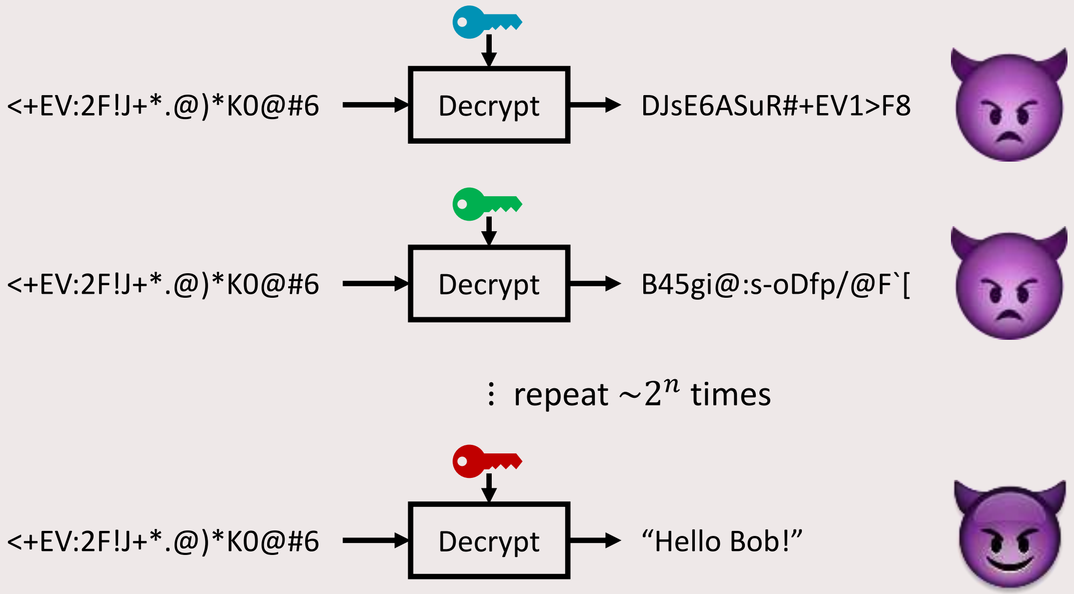
Encryption

- Symmetric encryption: Alice and Bob need to share a secret key
  - examples: AES, ChaCha20, one-time pad



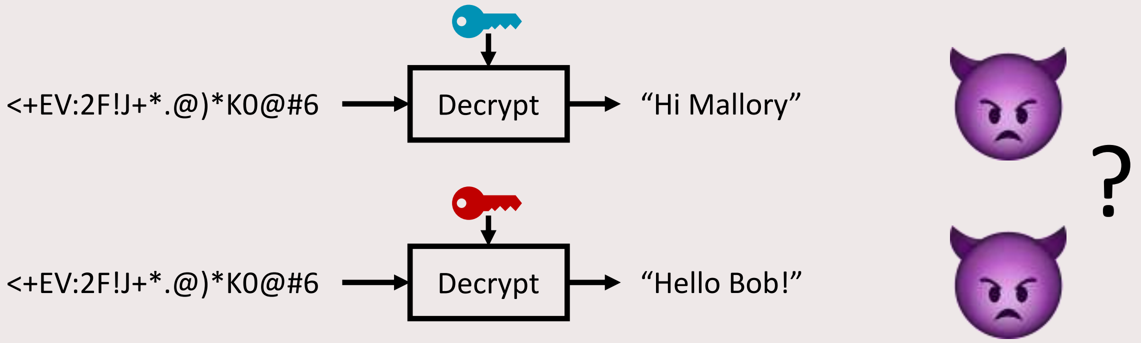
Confidentiality (computational)

- The ciphertext “gives no information” about the plaintext
- *n*-bit security: Mallory expects to try 2^n keys before finding the right one



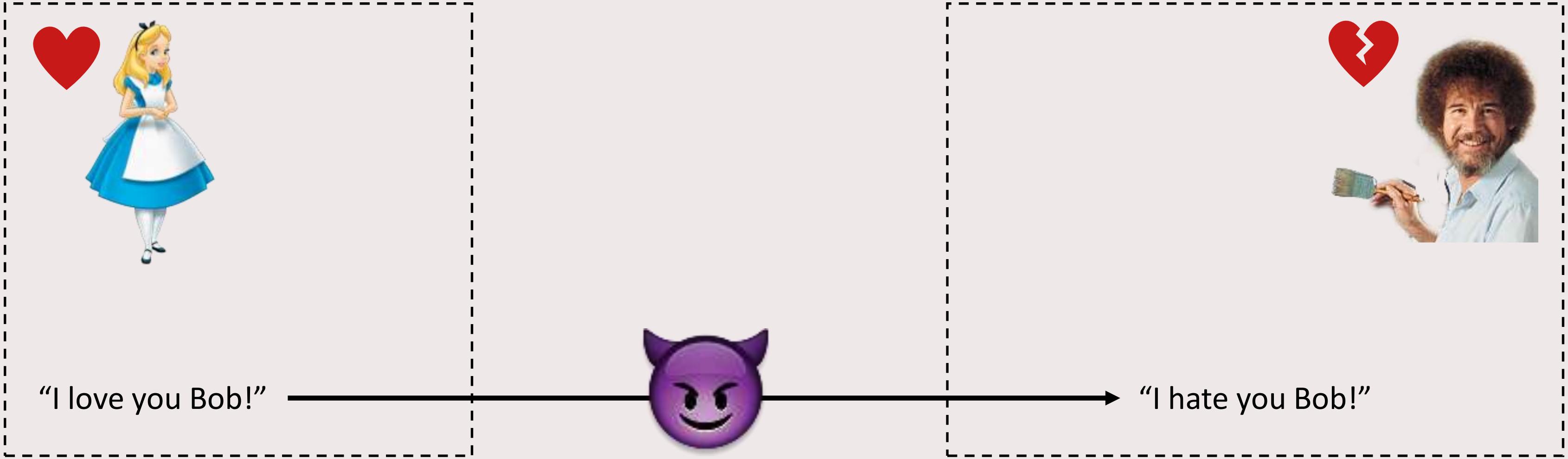
Confidentiality (information theoretical)

- Perfect security
  - Mallory has no way of distinguishing correct decryptions from incorrect ones
- Requires a **one-time** pad
  - Key can only be used once



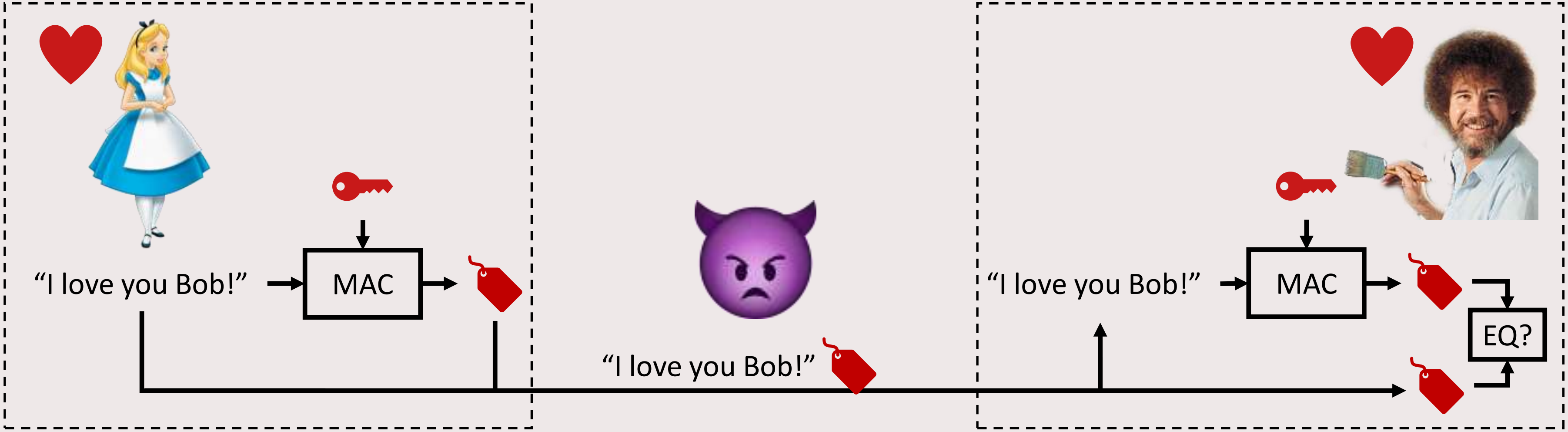
Integrity and authentication

- Integrity: nobody should be able to change the message
- Authentication: Bob knows the message came from Alice



Message Authentication Code (MAC)

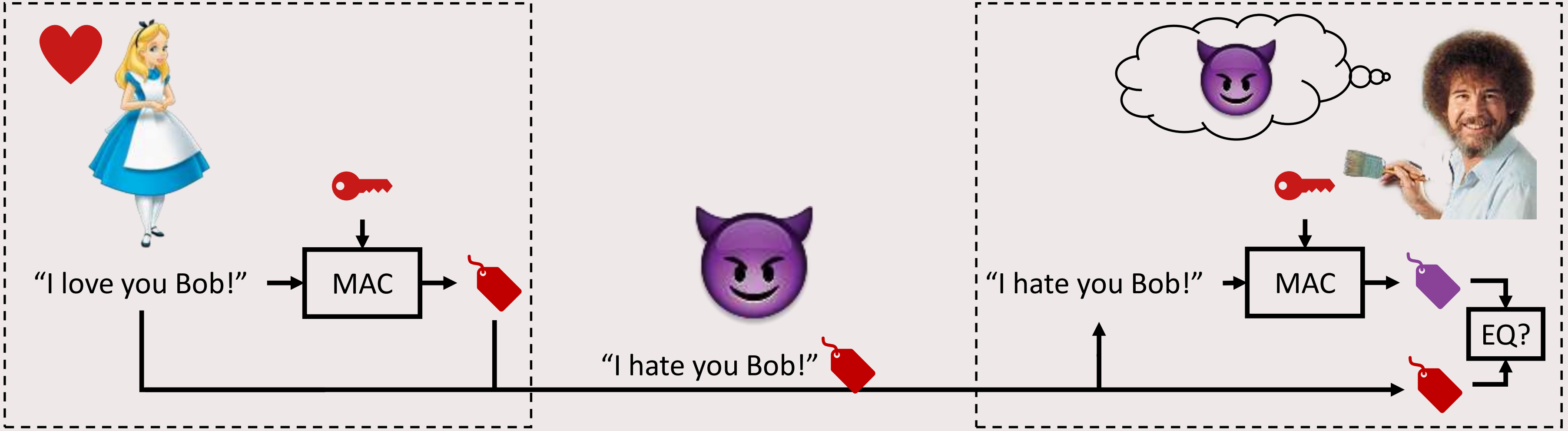
- Symmetric: Alice and Bob need to share a secret key
  - allows Bob to detect any changes
  - examples: HMAC, Poly1305



TU/e

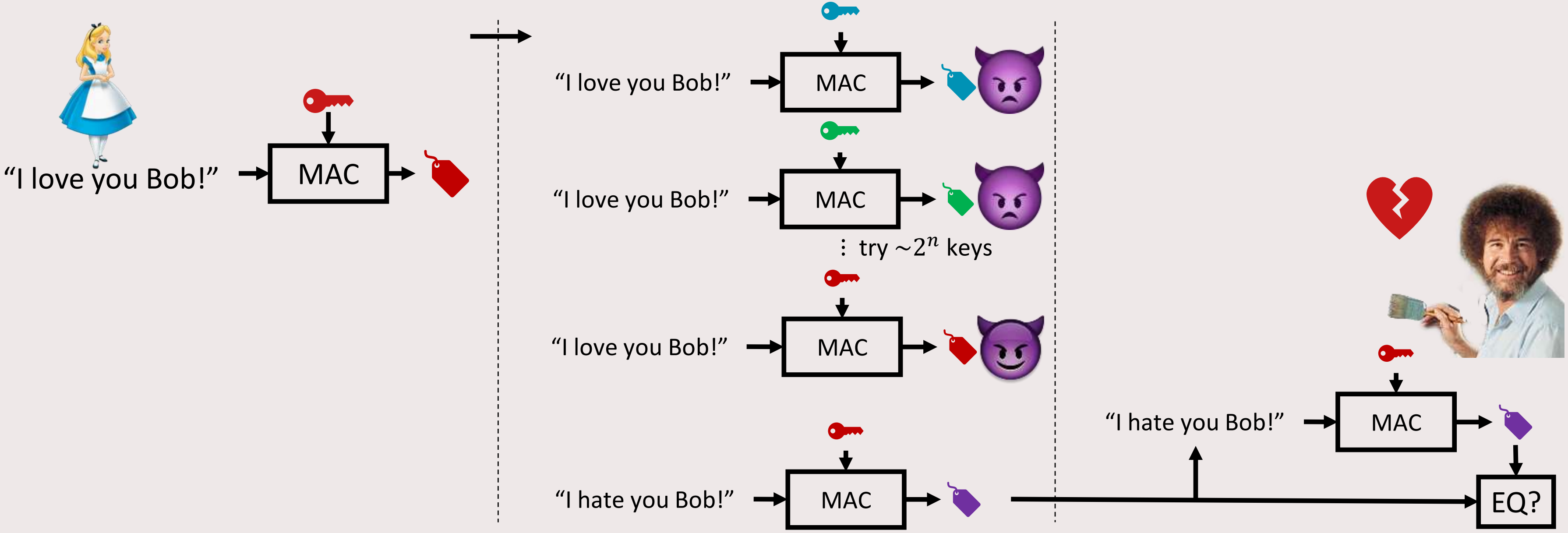
Message Authentication Code (MAC)

- Symmetric: Alice and Bob need to share a secret key
  - allows Bob to detect any changes
  - examples: HMAC, Poly1305



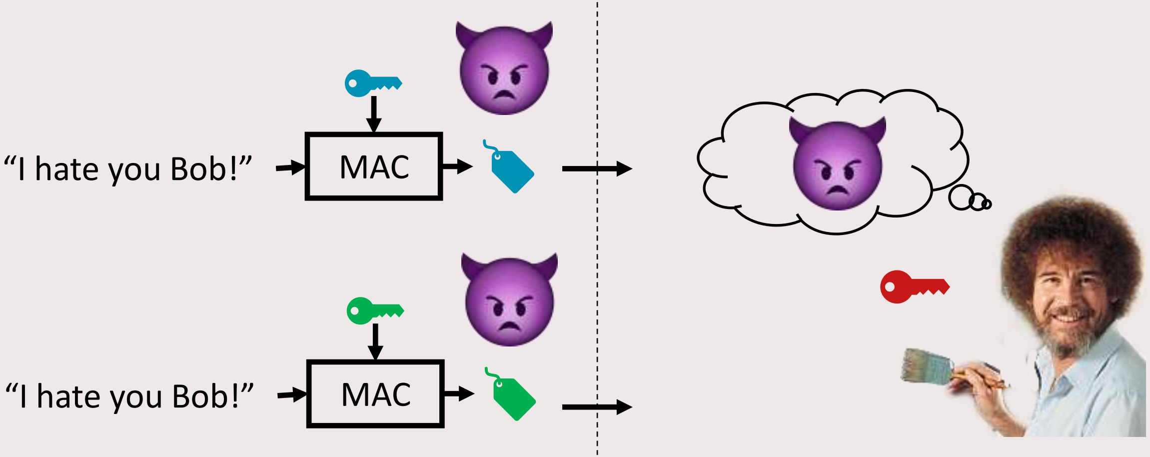
Unforgeability (computational)

- Mallory cannot forge tags for any (other) message
- *n*-bit security: Mallory can locally try to find the correct key among 2^n keys



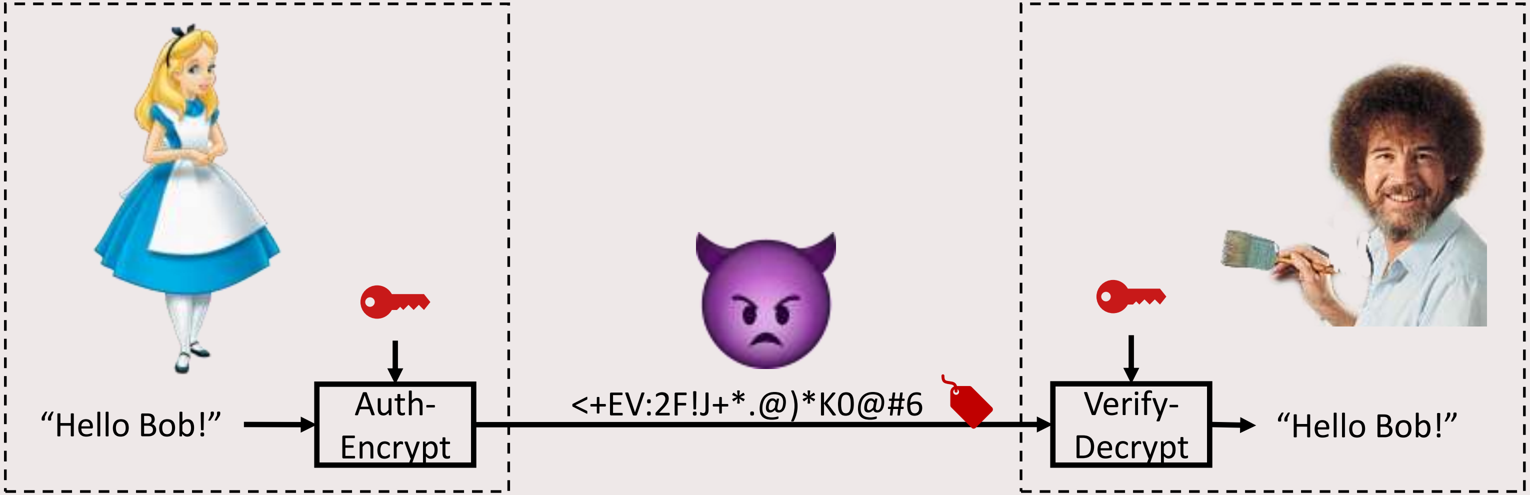
Authentication (information theoretical)

- Mallory cannot verify forgeries locally
- n-bit security: each forgery succeeds with probability 2^{-n}
  - *statistical* security
- Requires discarding the authentication key (or at least some part of it)
  - "encrypt" the tag with a one-time pad



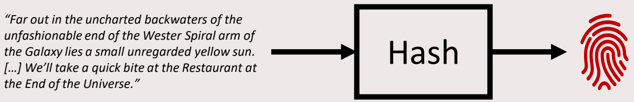
Authenticated Encryption

- Combined encryption and authentication
  - required for confidentiality against active attackers!



Cryptographic hashing

Given a long message *M*, a hash function computes a *small* message digest



The digest is also called the fingerprint, or simply "the hash of M".
Note there is no key involved.

Hash should behave as a random function:

- given , it should be hard to compute *M*
- it is hard to find any M_0, M_1 such that $\mathrm{Hash}(M_0) = \mathrm{Hash}(M_1)$

Hash functions are used everywhere in cryptography.
Examples: ~~MD5 (broken)~~, SHA2, SHA3

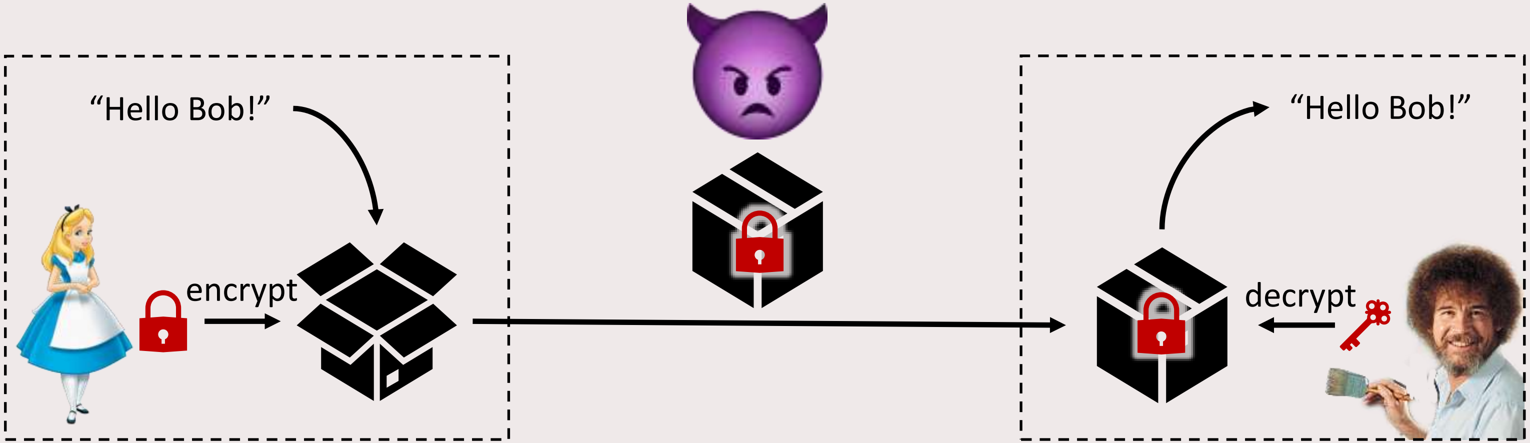
Public key cryptography

- Parties generate a keypair: (,)
  - give the public key ( ) to everybody, so anybody can use it
  - keep the private key ( ) secret
- Also called asymmetric cryptography
- Example usage:
  - key exchange
  - digital signatures
  - public key encryption
- Example systems, used on the internet today:
  - RSA
  - Elliptic curve cryptography (ECC)
  - Diffie-Hellman key exchange (DH)

The box analogy is better suited for hybrid authenticated encryption (e.g. crypto_secretbox in NaCl)

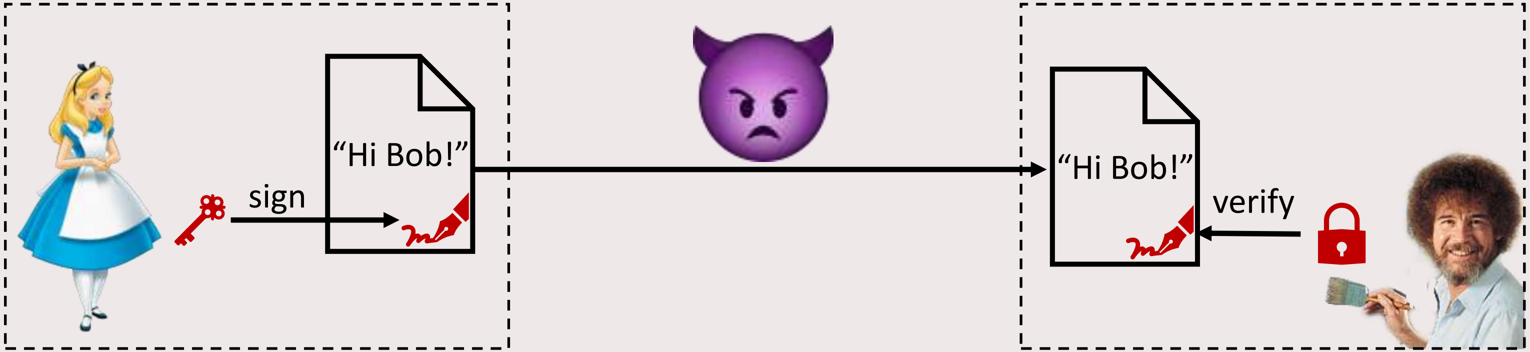
Public key encryption

- Bob generates a keypair: (,), gives to Alice
- Provides confidentiality, but no authenticity (because everybody can encrypt)



Digital signatures

- Alice generates a keypair: (🗝 , 🔒), gives 🔒 to Bob
  - Alice can put a signature (✒) on any message, using her private key (🗝)
  - Provides:
    - message integrity (nobody can change the message)
    - message authentication (Bob knows message came from Alice)
    - non-repudiation (Alice can't deny signing message)



Example: RSA

- Rivest-Shamir-Adleman (RSA)

- Private key (🗝): two random large primes (p, q)
- Public key (🔒): $N = p \cdot q$
- System parameter: e (usually 65537)

- Security based on the hardness of *factoring*
  - given N, it should be hard to find p, q

Example: RSA-KEM

- Key encapsulation mechanism (KEM)
  - generate a random symmetric key *k* ( )
  - (authenticate-)encrypt the message using *k*
  - encapsulate *k* to the recipient's public key (= *N*)

- Alice knows Bob's public key *N*:
  1. she generates a random *k*
  2. she encapsulates *k*:

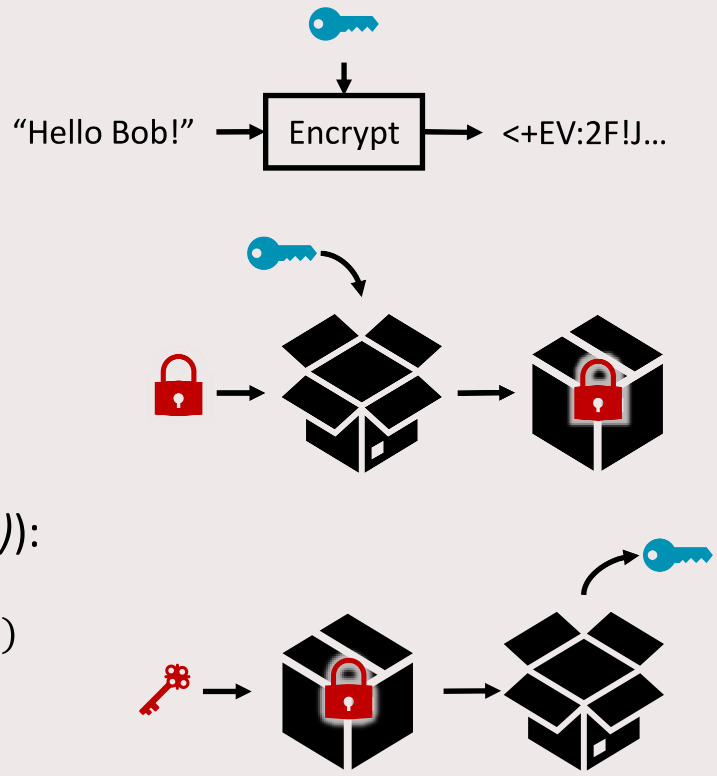
$$c = k^e \bmod N$$

- Bob, given *c* and using his private key (= *(p, q)*):
  1. he computes:

$$d = e^{-1} \bmod (p-1)(q-1)$$

  2. he *decapsulates*:

$$k = c^d \bmod N$$



Example: RSA signature

- Alice wants to sign message *M* using her private key (= *(p, q)*)
  1. she hashes the message

     $$= H = \mathrm{Hash}(M)$$

  2. she computes the signature

     $$\sigma = H^d \bmod N$$

- Bob verifies *(M, σ)* using Alice's public key (= *N*)
  1. he computes

     $$H' = \sigma^e \bmod N$$

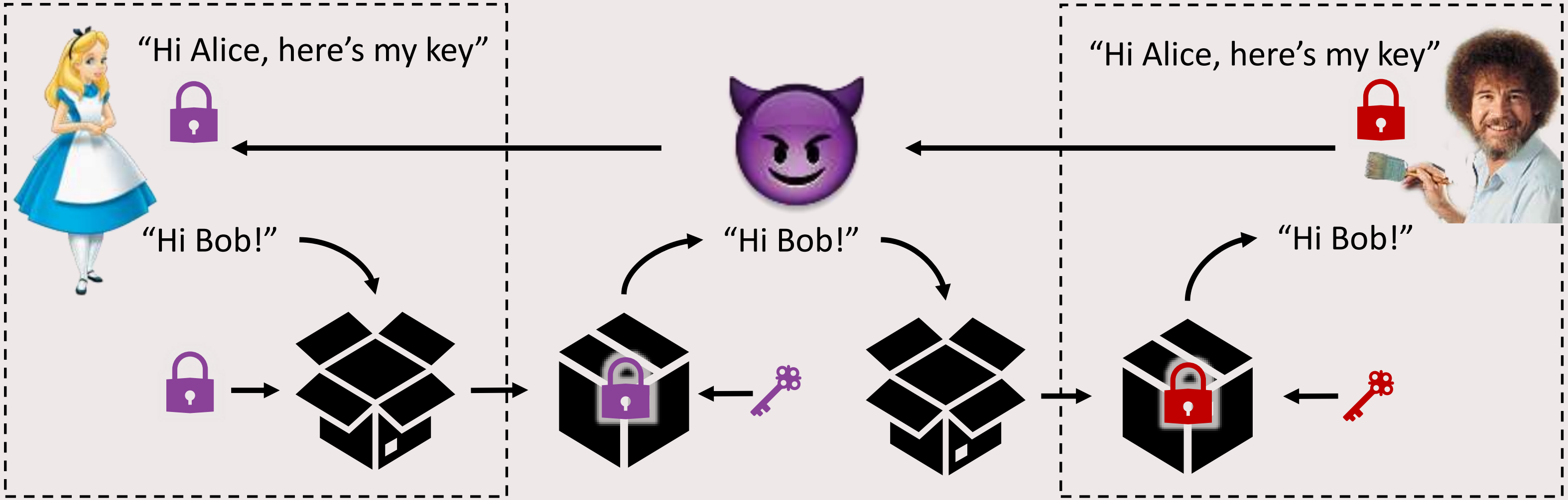
  2. he hashes the message

     $$H = \mathrm{Hash}(M)$$

  3. he checks if *H = H'*

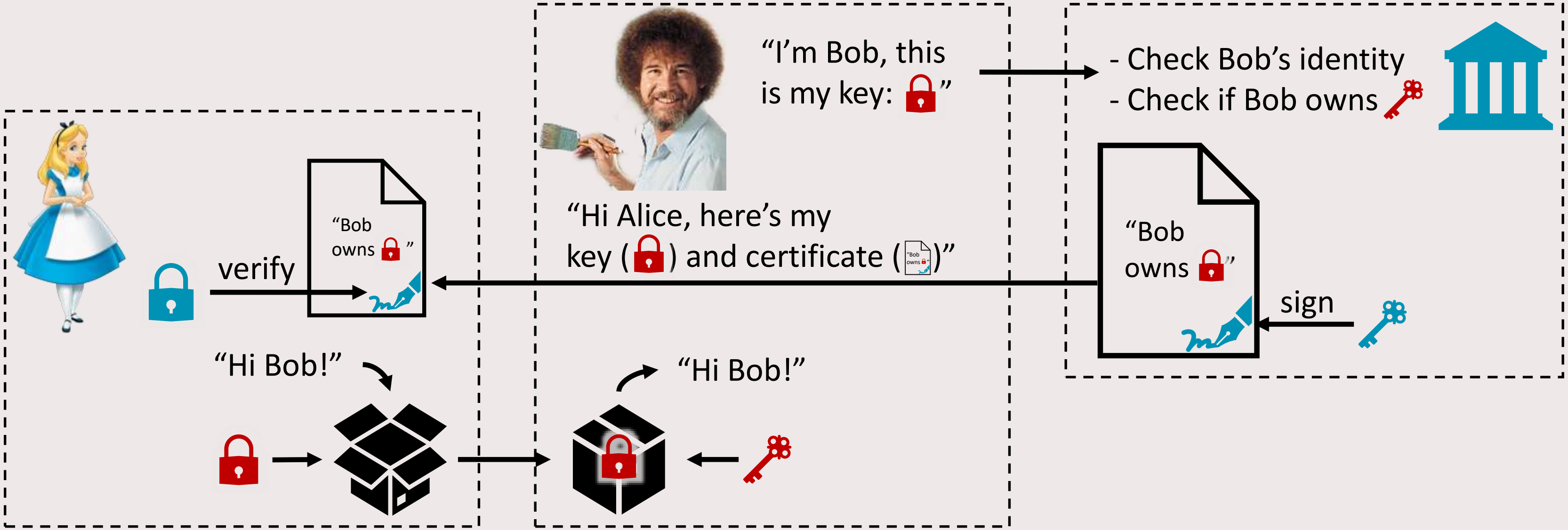
Key authentication

- Public keys are usually provided at the start of a protocol
- How do you know the key actually belongs to the claimed owner?
  - you need key authentication,
    otherwise you are vulnerable to a Mallory-in-the-Middle attack



Certificates

- Requires a trusted third party (🏛)
- Alice must have 🔒, for example pre-installed on her computer



Outline

- Cryptography
  - Basics
  - Post-Quantum Cryptography (PQC)
- Quantum Key Distribution (QKD)
  - QKD Network
  - TU/e testbed

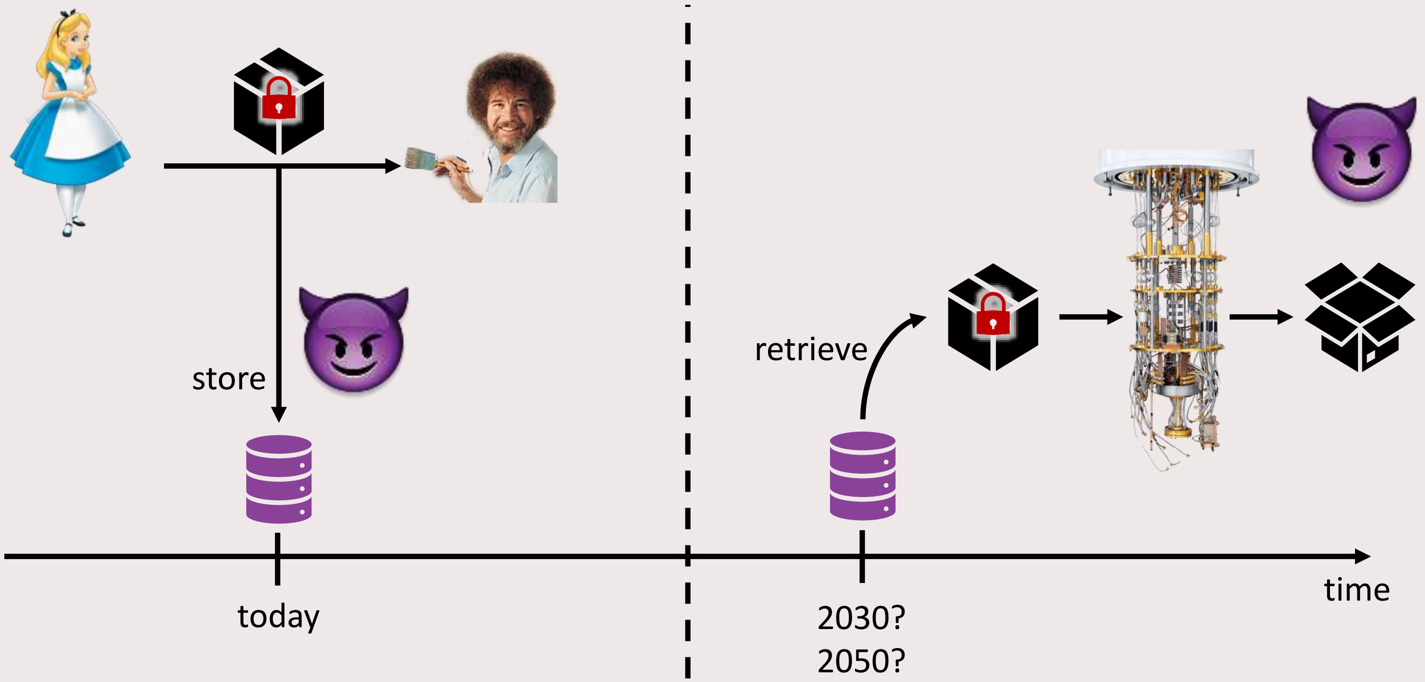
Quantum Computers

Two* algorithms threaten existing cryptography

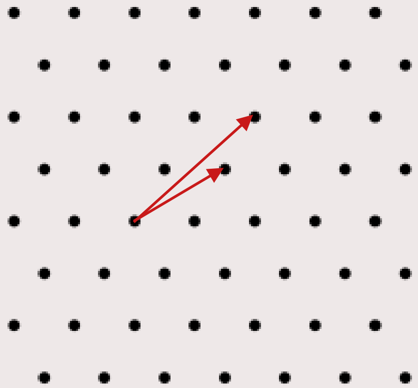
1. Shor's algorithm for period finding can efficiently …
   a. … factor N ⇒ **breaks RSA**
   b. … find discrete logarithms ⇒ **breaks ECC, breaks DH**
2. Grover's search algorithm can …
   - ... try 2^n keys with only $2^{n/2}$ quantum queries
     ⇒ double key-length suffices for symmetric cryptography

*) More algorithms exist, but their impact on widely deployed cryptography is roughly the same as Grover's algorithm.

Harvest now, decrypt later



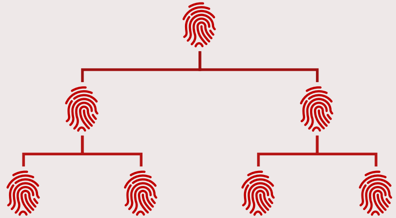
Post-Quantum Cryptography (PQC)



Alice & Bob have classical computer
Mallory has a quantum computer

Replace factoring (or discrete log) with other problems:

- Lattice-based cryptography
  - both KEMs and signatures
- Hash-based cryptography
  - signatures
- Error correcting codes
  - KEMs
- Multivariate cryptography
  - (mainly) signatures
- Isogeny-based cryptography (maybe broken?)

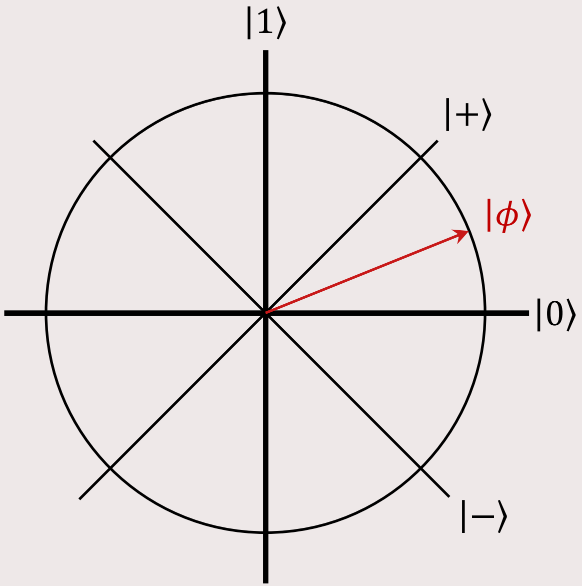


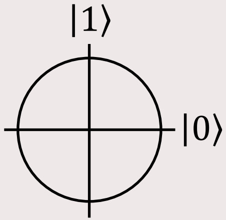
Outline

- Cryptography
  - Basics
  - Post-Quantum Cryptography (PQC)
- Quantum Key Distribution (QKD)
  - QKD Network
  - TU/e testbed

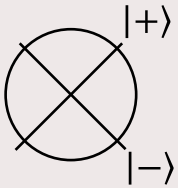
Quantum information (the bare minimum for QKD)

A *qubit* is a vector





computational basis (+)

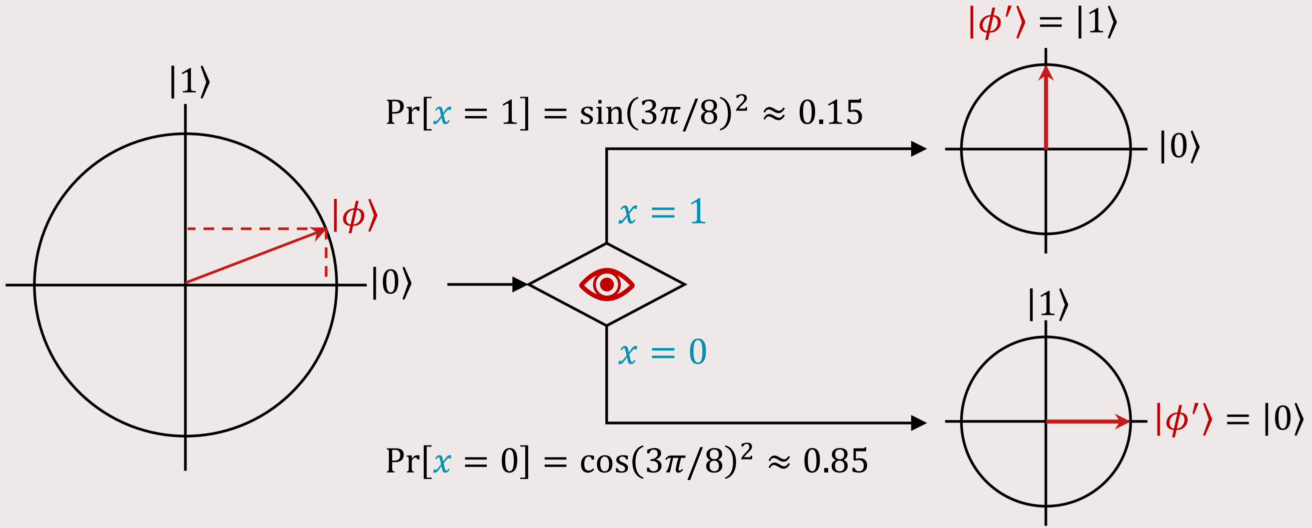


Hadamard basis (×)

Measurement

If we **measure** ( ) a qubit

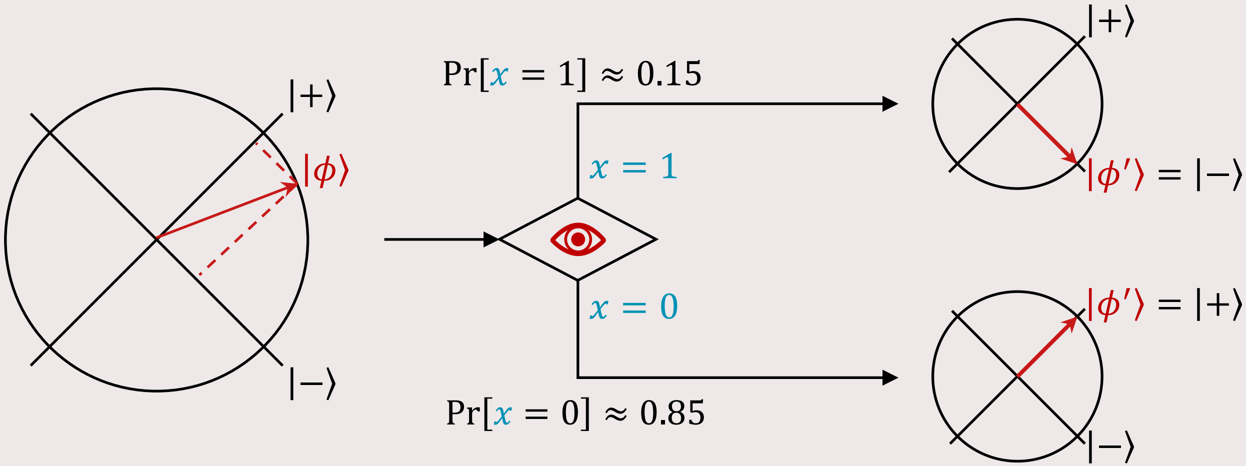
- it *collapses* onto the measurement basis
  - with probability defined by the in-product of qubit and basis vector
- we get a classical bit (x) as output



Measurement

If we **measure** ( ) a qubit

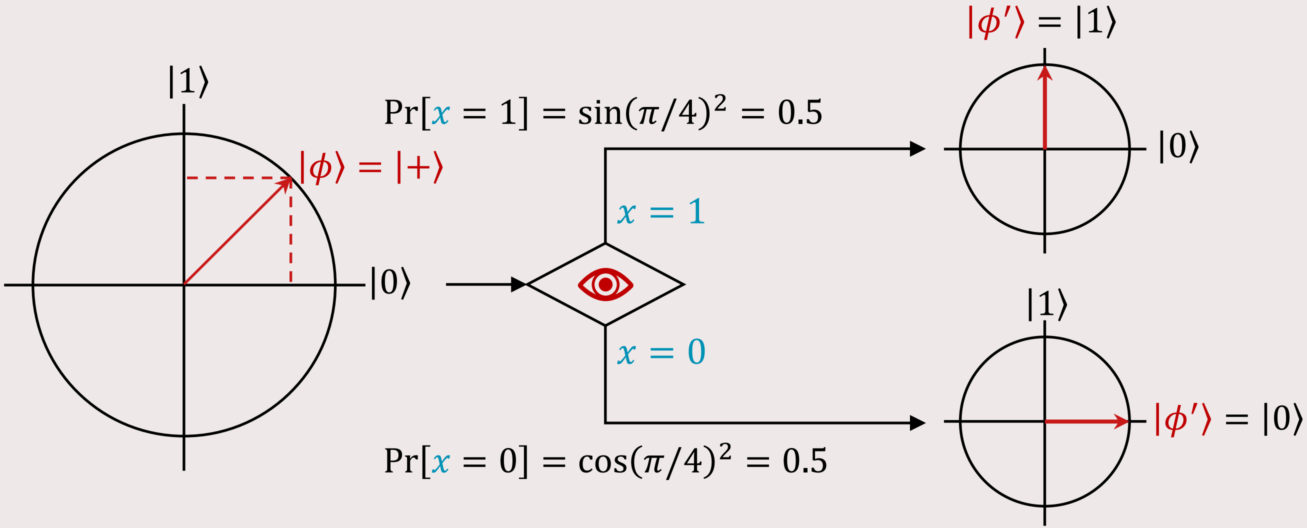
- it *collapses* onto the measurement basis
  - with probability defined by the in-product of qubit and basis vector
- we get a classical bit (x) as output



Measurement

If we **measure** (eye icon) a qubit

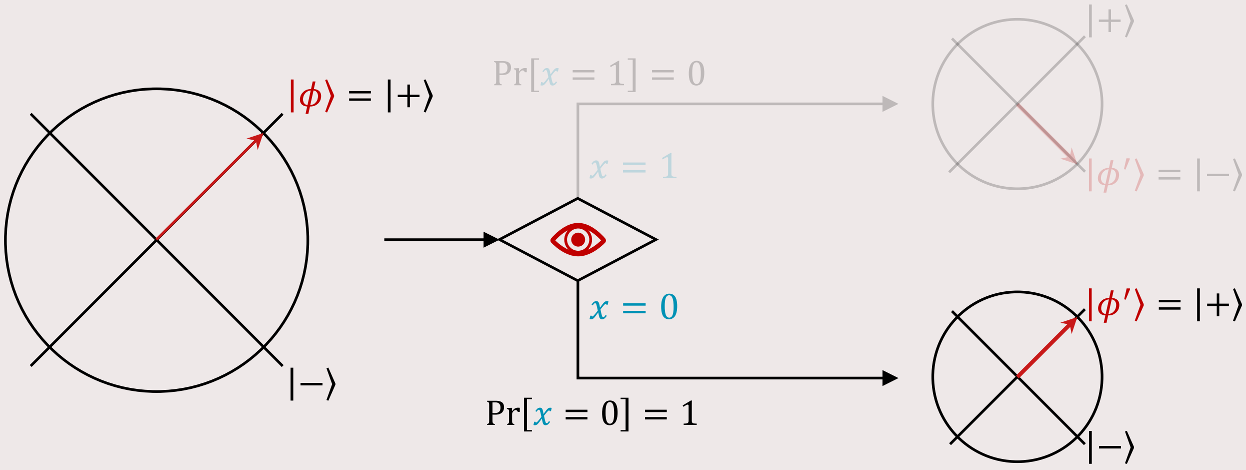
- it *collapses* onto the measurement basis
  - with probability defined by the in-product of qubit and basis vector
- we get a classical bit (x) as output



Measurement

If we **measure** (👁) a qubit

- it *collapses* onto the measurement basis
  - with probability defined by the in-product of qubit and basis vector
- we get a classical bit (x) as output



Bennett-Brassard (BB84)

1.	$\vert+\rangle$	$\vert1\rangle$	$\vert-\rangle$	$\vert0\rangle$	$\vert1\rangle$	$\vert1\rangle$	$\vert0\rangle$	$\vert0\rangle$	$\vert-\rangle$	$\vert+\rangle$	$\vert1\rangle$	$\vert-\rangle$	$\vert+\rangle$	$\vert+\rangle$	$\vert1\rangle$
2.	$+$		$\times$		$+$	$\times$	$\times$	$+$		$+$	$\times$	$\times$		$\times$	$+$
3.			✓		✓			✓				✓		✓	
4.					1									0	
5.					✓									✓	
6.			**1**					**0**				**1**			**1**

1. Alice sends **random** qubits (some may not arrive)
2. Bob measures in **random** bases, reveals them to Alice **after** the measurement
3. Alice confirms when sending/measurement basis were the same
4. Bob reveals each measurement outcome bit with probability ½
5. Alice confirms the bits are correct (and aborts if any bit is incorrect)
6. Both use the remaining bits as shared key: 1011

All classical messages are authenticated, as indicated by the tags ( ).

BB84, improvements

- Information reconciliation
  - error *correction* instead of error detection
- Privacy amplification
  - Mallory may have some information about the secret bits
  - “distill” these bits a shorter key so Mallory has only negligible information
- Require fewer check bits

Security of QKD

- Key is statistically independent from Mallory’s observations
  - cannot be broken by trying more keys or future cryptanalysis
  - can be broken by exploiting discrepancies between hardware and model

- Use key as one-time pad + statistical MAC:
  - security independent of any computational assumptions
- Use key in computational (symmetric) cryptography
  - breaks only if the computational cryptography breaks
  - (this is often done because of the low key-rate of QKD)

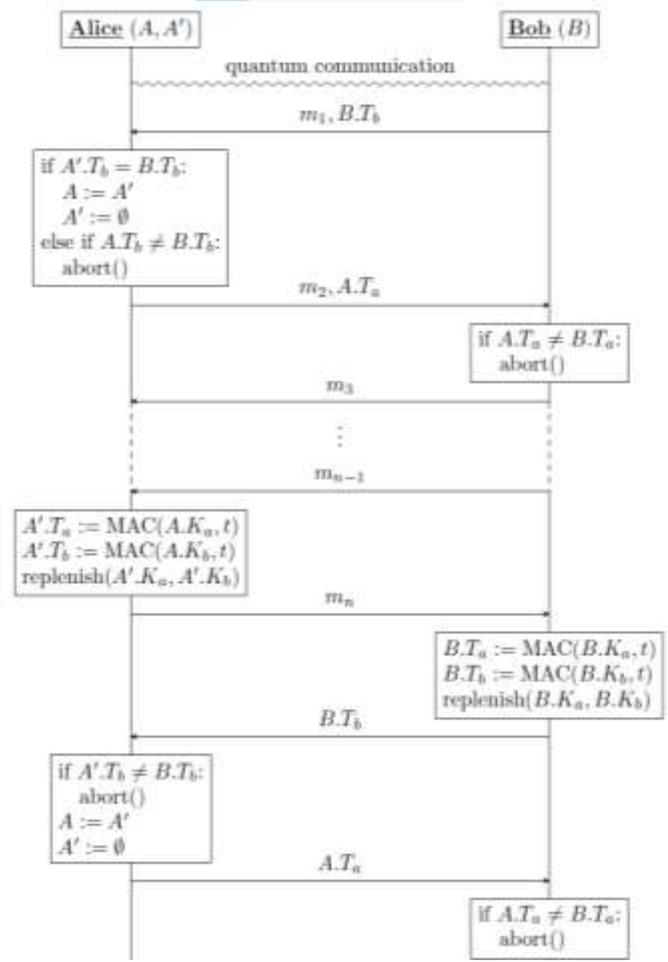
QKD authentication

- Authentication typically done with statistically secure MACs
  - but then we assume shared keys
    - so it’s not key *distribution*, so much as it is key *expansion*
  - and we have to discard some key material
    - consumed keys can be replaced with fresh QKD output
    - requires some care to prevent key exhaustion (by Mallory)

- However, we can authenticate with computational MACs or signatures
  - **if** authentication isn’t broken now,
    then the output key will never be broken later

Preventing key exhaustion

- Authenticate every message computationally
- Authenticate the *transcript* statistically
- Prevent Mallory from desynchronizing us:
  - Compute tags the round before sending them
  - Send the previous tags at the session start



Outline

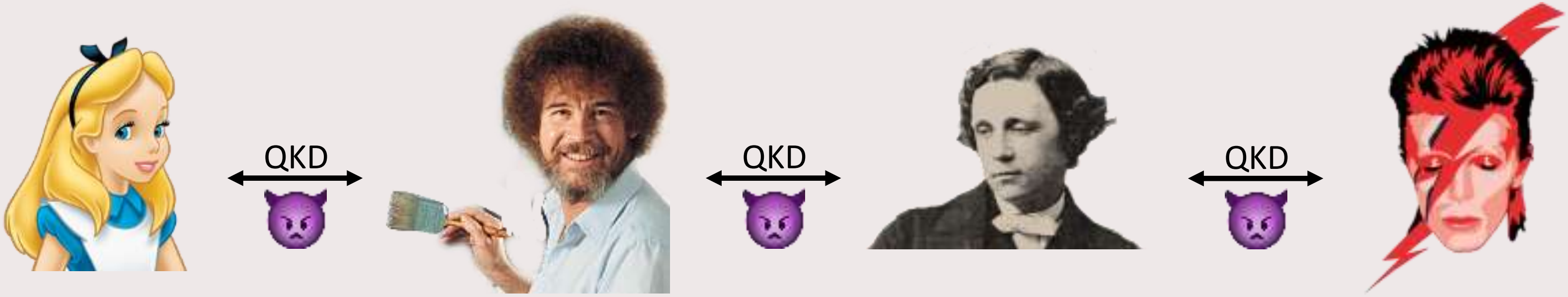
- Cryptography
  - Basics
  - Post-Quantum Cryptography (PQC)
- Quantum Key Distribution (QKD)
  - QKD Network
  - TU/e testbed

QKD limitation

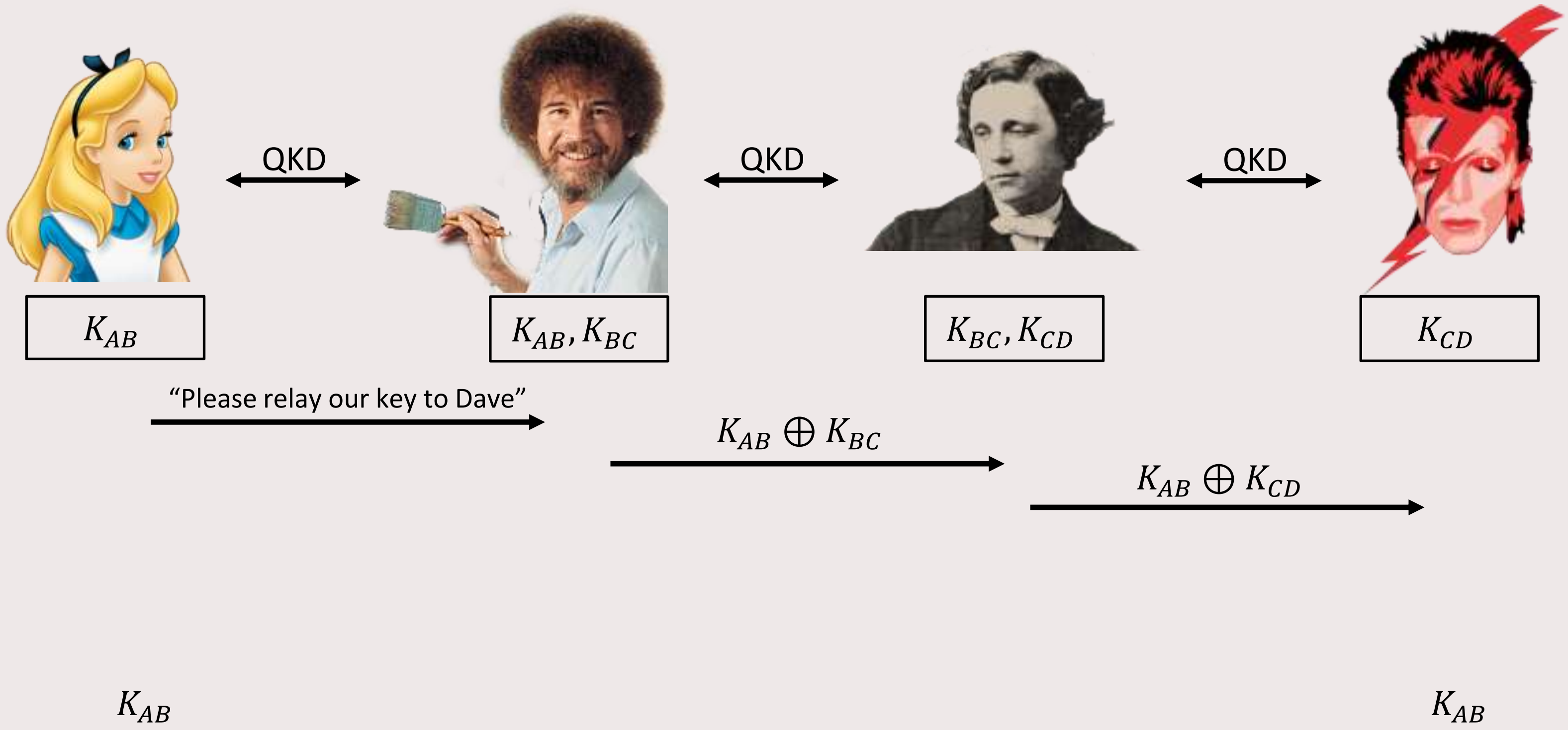
- QKD is a point-to-point protocol
- Single photon travel distance in fiber/free-space is limited
  - up to hundreds of kilometer (<100 km in practice)
  - but key-rate drops with larger distance
- No repeaters allowed
  - You cannot measure and resend the qubits (for the same reason Mallory can’t)
  - quantum repeaters theoretically exist, but require stable quantum memory

Trusted repeater network

- Meet Alice, Bob, Carroll, and David
- Each neighbouring pair is linked via QKD
- They trust each other, which means …
  - … they follow the protocol specification
  - … *throw away keys* after they have been used
  - … take care of their devices and keep out hackers/three letter agencies



Key relay



QKD
QKD
QKD
K_{AB}
K_{AB}, K_{BC}
K_{BC}, K_{CD}
K_{CD}
"Please relay our key to Dave"
$K_{AB} \oplus K_{BC}$
$K_{AB} \oplus K_{CD}$
K_{AB}
K_{AB}

Key relay

QKD ↔

QKD ↔

QKD ↔

K_{AB}	K_{AB}, K_{BC}	K_{BC}, K_{CD}	K_{CD}

"Please relay our key to Dave" →

$K_{AB} \oplus K_{BC}$ →

$K_{AB} \oplus K_{CD}$ →

$k_A, ct_A = \text{KEM}(pk_D)$

$k_D, ct_D = \text{KEM}(pk_A)$

ct_A →

← ct_D

$K_{AB} \oplus \text{KDF}(k_A, k_D)$

$K_{AB} \oplus \text{KDF}(k_A, k_D)$

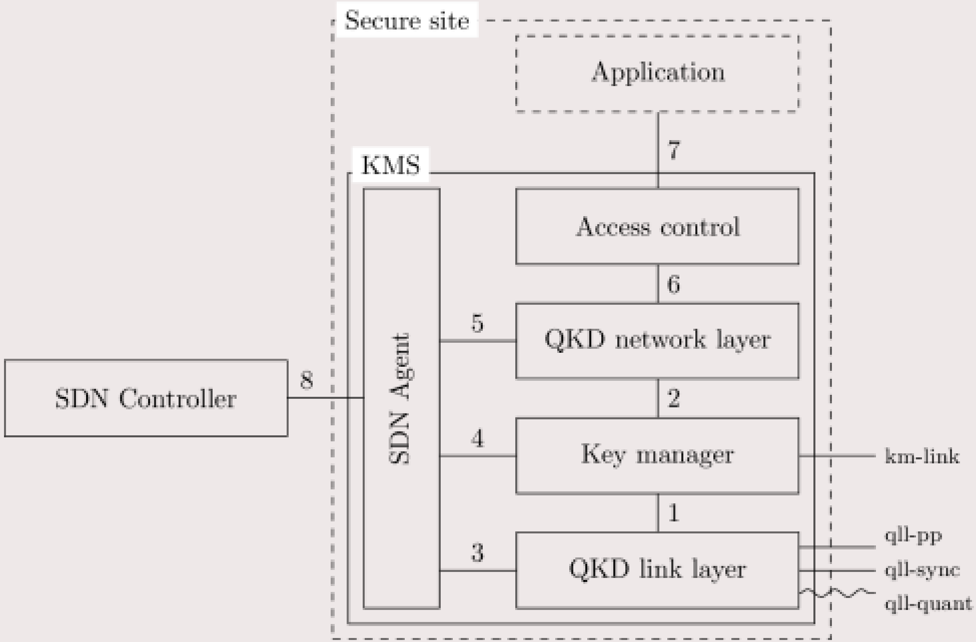
Outline

- Cryptography
  - Basics
  - Post-Quantum Cryptography (PQC)
- Quantum Key Distribution (QKD)
  - QKD Network
  - TU/e testbed

Key Management Server

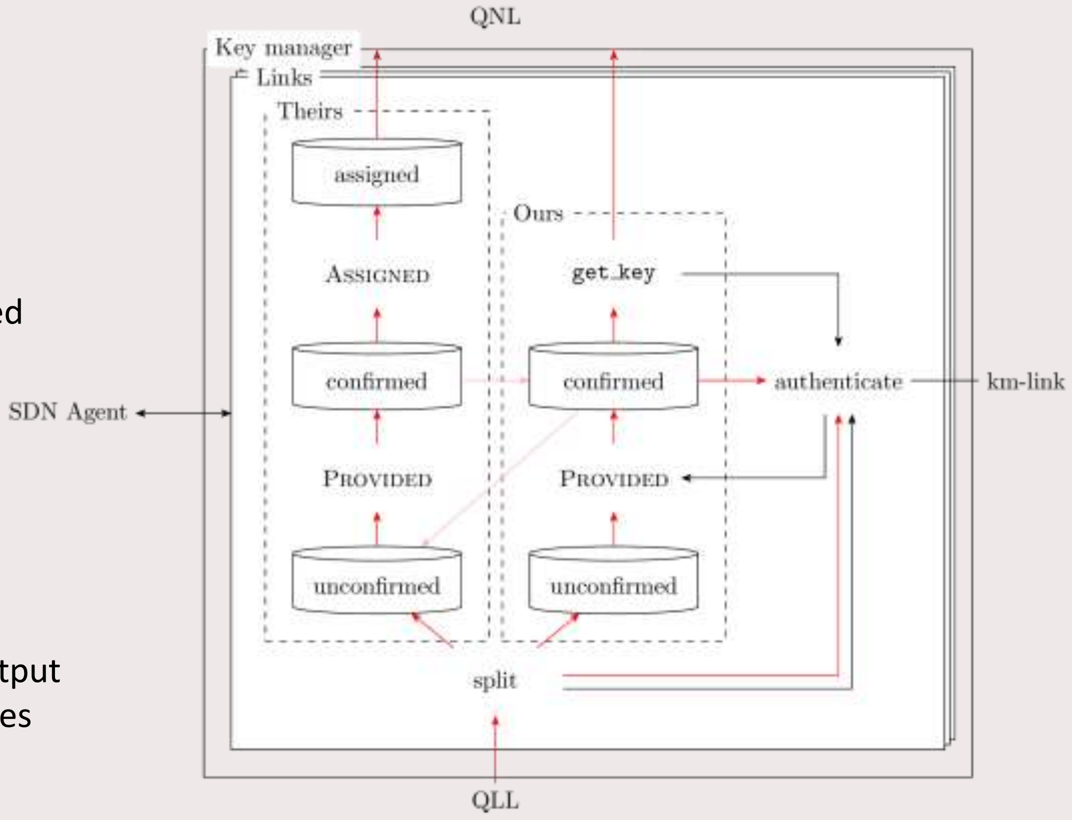
- Apps live within secure site

- QLL manages QKD protocols
- KM caches keys
- QNL relays between neighbours
- SDN determines routes



Key manager

- KM Main goal: synchronizing keys
  - incoming keys are split ~50/50
  - we can always take from `ours`
  - we only take from `theirs` if instructed
- Multiple links per node
  - Bob is linked to Alice
  - Bob is linked to Carroll
- Multiple providers per link
  - Alice and Bob may run multiple QKD protocols to increase bandwidth
- Authenticate using MACs
  - use fresh key for confirming fresh output
  - use confirmed keys for other messages



Eindhoven QKD testbed – phase 1

TU/e

Eindhoven QKD testbed – phase 2

Eindhoven QKD testbed – phase 3

Phase 4?

Thank you

Slides are available online:

https://zeroknowledge.me/talks/#iotalentum22

s.r.verschoor@tue.nl

Quantum information (slightly beyond the bare minimum)

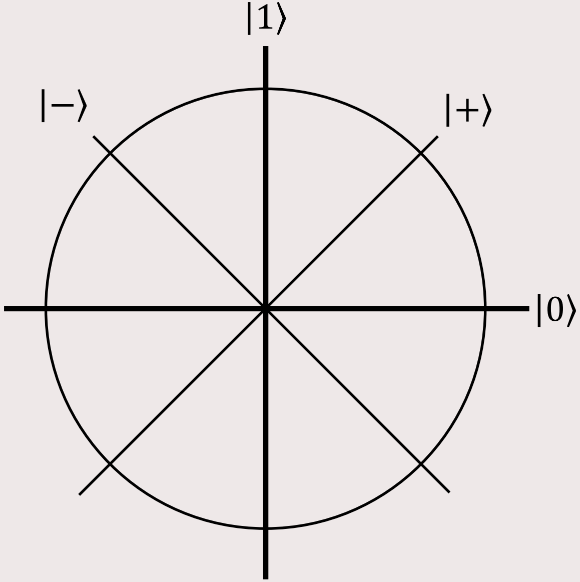
A qubit is a binary state of a quantum system

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$|+\rangle = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

$$|-\rangle = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$



Generally $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, with $|\alpha|^2 + |\beta|^2 = 1$

Quantum information (slightly beyond the bare minimum)

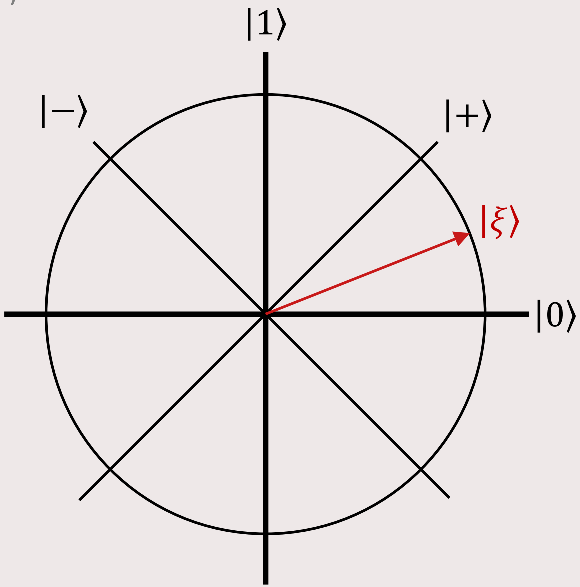
The *dual* vector of $|\psi\rangle$ is $\langle\psi| = (\alpha^*, \beta^*)$ (the conjugate transpose)
Then $\langle\phi|\psi\rangle = \langle\phi| \cdot |\psi\rangle$ is an inner product.

If we **measure** $|\psi\rangle$ in *computational* basis $\{|0\rangle, |1\rangle\}$,
then $|\psi\rangle$ **is destroyed** and we get an output label x:

$$\Pr[x = 0] = |\langle 0|\psi\rangle|^2$$

$$\Pr[x = 1] = |\langle 1|\psi\rangle|^2$$

Similarly if we measure in *Hadamard* basis $\{|+\rangle, |-\rangle\}$:
$\Pr[x = 0] = |\langle +|\psi\rangle|^2$ and $\Pr[x = 1] = |\langle -|\psi\rangle|^2$



Example: if we measure $|\xi\rangle$ (see picture) in either basis,
we get output label 0 with probability $\frac{2+\sqrt{2}}{4} \approx 0.85$

TU/e

Key relay (alternative)

QKD

QKD

QKD

K_{AB}	K_{AB}, K_{BC}	K_{BC}, K_{CD}	K_{CD}

$K_R \leftarrow$

$K_R \oplus K_{AB}$

$K_{AB} \oplus K_{BC}$

$K_{BC} \oplus K_{CD}$

K_R

$$K_R = (K_R \oplus K_{AB}) \oplus (K_{AB} \oplus K_{BC}) \oplus (K_{BC} \oplus K_{CD}) \oplus K_{CD}$$