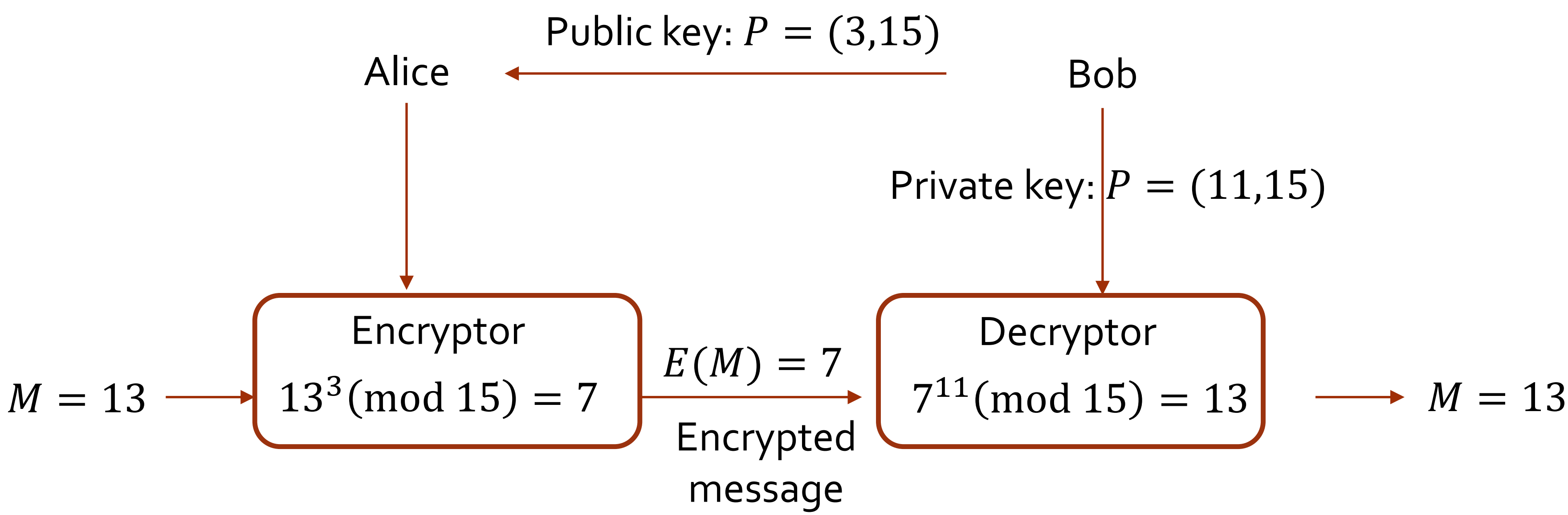
Review of Quantum Cryptography

- Symmetric key system
  - Quantum key distribution system
- Public key system
  - Example: RSA (Rivest–Shamir–Adleman) public-key cryptosystem
  - The security of RSA is guaranteed by the difficulty of factoring a large number ➔ RSA factoring challenge
  - Also very useful for authentication

Example of RSA encryption/decryption

- message to transmit: M
- Encryption: $E(M) = M^e (\mathrm{mod}\ n)$
- Decryption: $D\big(E\ (M)\big) = E(M)^d (\mathrm{mod}\ n)$

- Public key: $P = (e, n) = (3, 15)$
- Private key: $S = (d, n) = (11, 15)$
- Assume that the intended message M is 13.
- Encryption: $E(13) = 13^3 (\mathrm{mod}\ 15) = 7$
- Decryption: $D\big(E\ (13)\big) = 7^{11} (\mathrm{mod}\ 15) = 13$



- In fact, $M^{e \cdot d} (\mathrm{mod}\ n) = M^{33} (\mathrm{mod}\ 15) = M$ for $0 \leq M < 15$

Example of RSA key generation

- Generation of public/private keys for RSA
  - Select two large prime numbers, p and q.
  - Compute the product $n \equiv pq$.
  - Select at random a small odd integer, e, that relatively prime to $\phi(n) = (p-1)(q-1)$.
  - Compute d, the multiplicative inverse of e, modulo $\phi(n)$.
  - The RSA public key is the pair $P = (e, n)$. The RSA private key is the pair $S = (d, n)$.

- $p = 3$ and $q = 5$.
- $n \equiv pq = 15$.
- $\phi(n) = (p-1)(q-1) = 8$

  $e = 3$
- $d = 11$ ➔ $e \cdot d = 33$

  ➔ $e \cdot d \left(\mathrm{mod}\ \phi(15)\right) =$

  $33 \pmod 8 = 1$
- RSA public key: $P = (3{,}15)$

  RSA private key: $S = (11{,}15)$

Summary of RSA

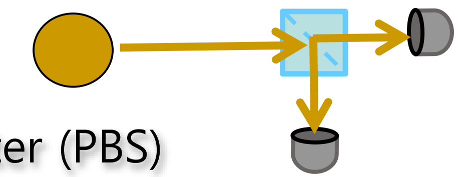
- Appendix 4.1, 4.2, 5
- Generation of public/private keys for RSA
  - Select two large prime numbers, p and q.
  - Compute the product $n \equiv pq$.
  - Select at random a small odd integer, e, that is relatively prime to $\phi(n) = (p-1)(q-1)$. $\phi(n)$ is called Euler('s totient) function.
  - Compute d, the multiplicative inverse of e, modulo $\phi(n)$.
  - The RSA public key is the pair $P = (e, n)$. The RSA private key is the pair $S = (d, n)$.
- Assume that the length of message M is $\mathrm{floor}(\log_2 n)$ bits.
- Encryption: $E(M) = M^e (\mathrm{mod}\ n)$
- Decryption:$D\big(E\ (M)\big) = E(M)^d (\mathrm{mod}\ n)$
- Sketch of proof that the above procedure will recover M
  - Assume: M is relatively prime (or co-prime) to n.
  - $D\big(E\ (M)\big) = E(M)^d (\mathrm{mod}\ n) = M^{ed} (mod\ n) = M^{1+k\phi(n)} (mod\ n)$
  - $= M \cdot M^{k\phi(n)} (mod\ n) = M (mod\ n)$
  - $M^{k\phi(n)} (mod\ n) = 1 (mod\ n)$ 🡨 Theorem A4.9

QRNG

- Quantum Random Number Generator (QRNG)
  - Basic concept: use the property of the quantum superposition to generate a true random number

$$|\psi\rangle = [|H\rangle - |H\rangle]/\sqrt{2}$$

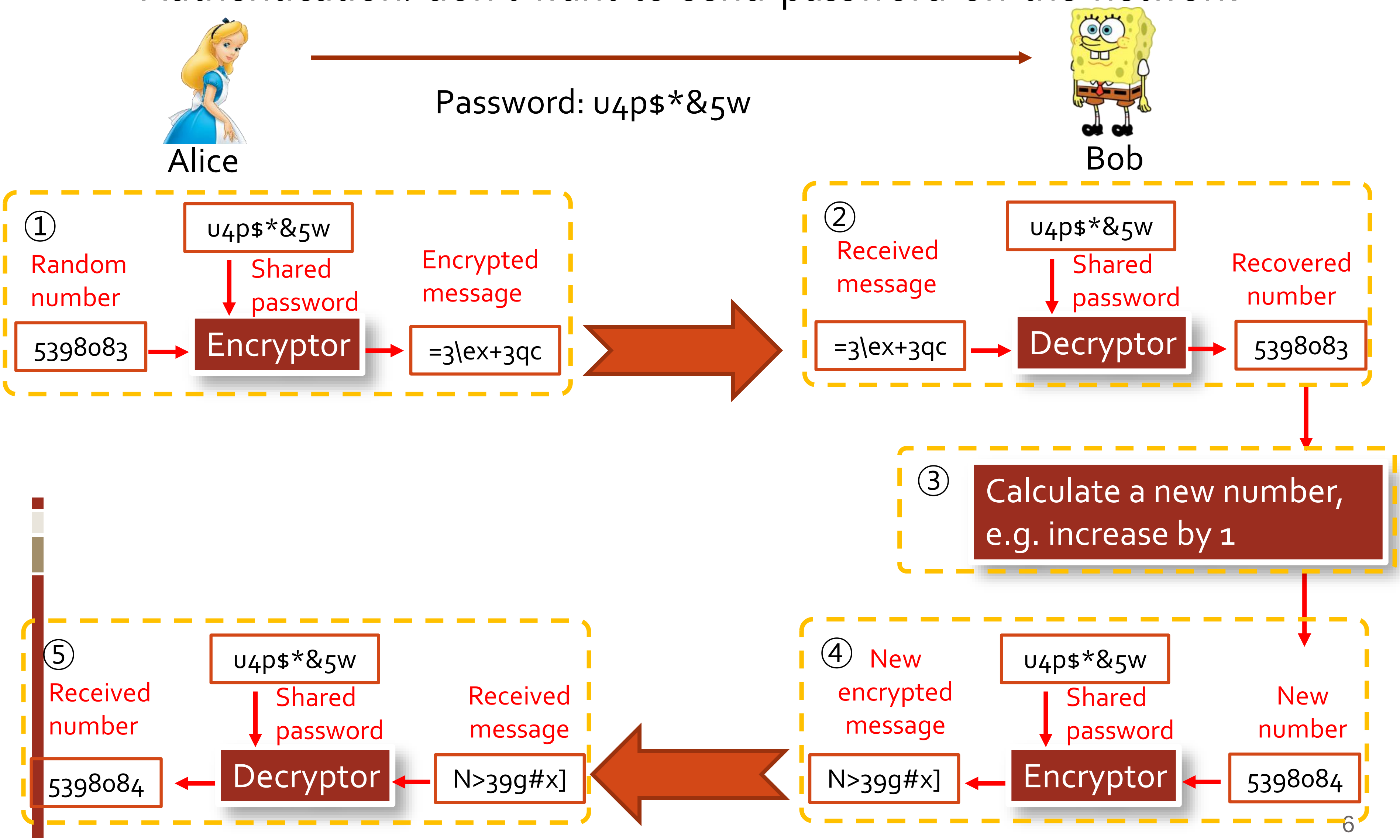
 polarization beam splitter (PBS)



  - Is this equivalent to QKD?
    - No! It has nothing to do with QKD.
- Pseudo random number generator (PRNG)
  - Random number is generated by complex calculation, starting with random seed number
  - If the same seed number is used, we can always predict the same random number
  - Seed number is generally supplied by non-repeating process such as current time or movement of computer mouse
  - However these seed numbers are relatively easily guessed ➔ We need perfect random number generator

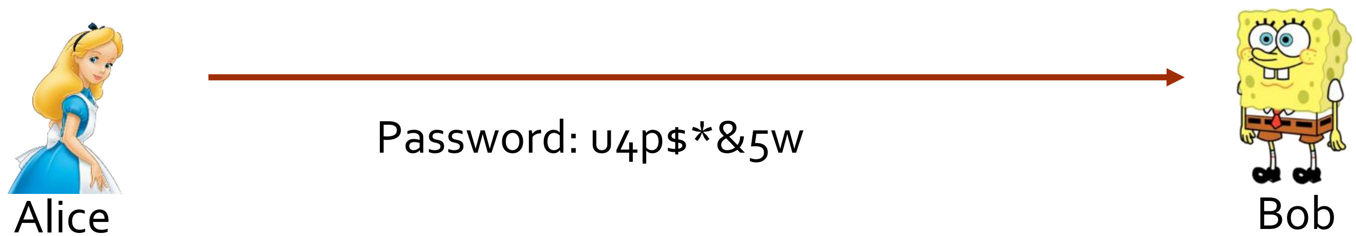
How to prove who you are?

- Authentication: don't want to send password on the network



Example Protocol for QRNG

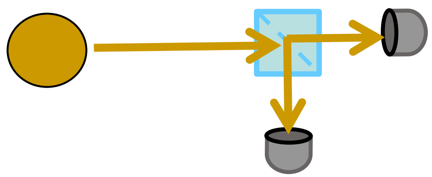
- Authentication: don't want to send password on the network

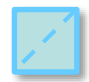


- Challenge-response
    1. Alice encrypts a random number with shared password and sends the encrypted message
    2. Bob decrypts the encrypted message using the mutually shared password
    3. Bob calculates new value based on the delivered random number
    4. Bob send back this new value encrypted again by the same password
    5. Alice decrypts the return message & verify Bob has the same password as well

Different Approaches to Implement QRNG Chip

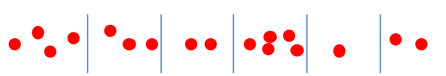
- What type of superposition can we use?
  - Superposition of paths





Beam splitter

  - Superposition of photon numbers



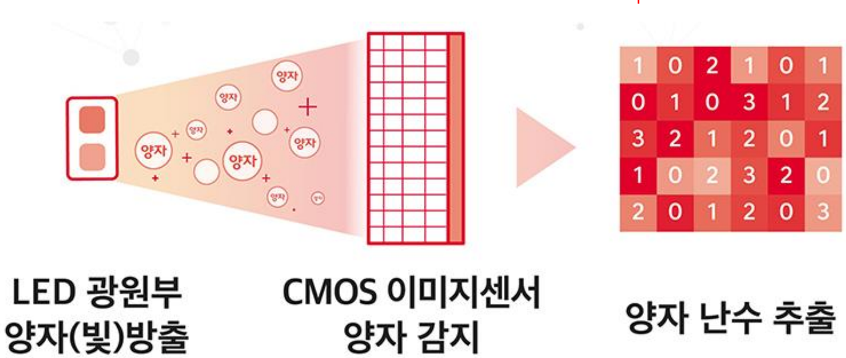
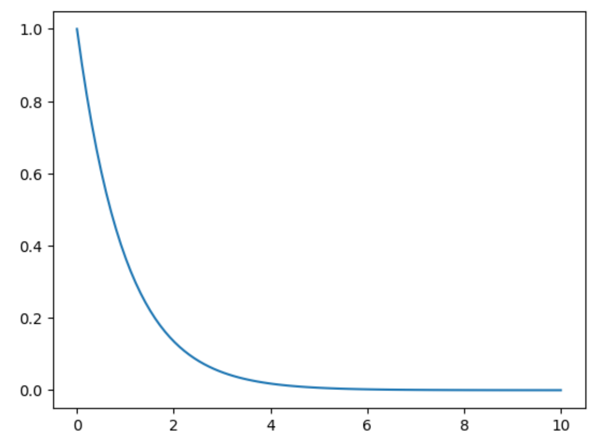


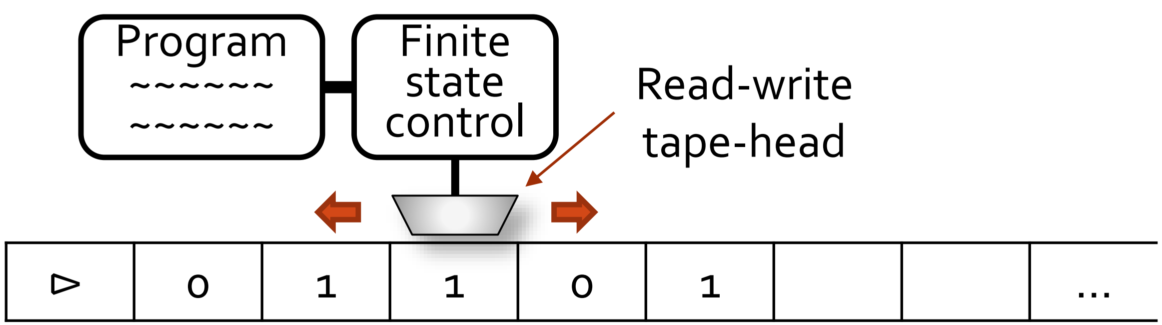
Image from https://www.sktinsight.com/123142

  - Superposition of times
    - Radioactive decay



Models for computation

- Elements of Turing machine (section 3.1.1)
  - Program
  - Finite state control
  - Tape (like a computer memory)
  - Read-write tape-head
- Finite state control consists of a finite set of internal states, $q_1, \dots, q_m$. q_s is a starting state and q_h is a halting state.
- Each square in the tape contains one symbol drawn from some alphabet Γ, a finite set of distinct symbols. $\triangleright$ indicates the left edge of the tape, and b means blank.
- The read-write tape-head starts from $\triangleright$ and after reading the symbol written in current square, update the value and move the head according to the program.



Turing machine

- A program for a Turing machine is a finite ordered list of program lines of the form $\langle q, x, q', x', s\rangle$. q is a current state and x is a value in the current square. q' is the next state and x' is the value to be written in the square. s is the direction to move the head.
- Example program

  1: $\langle q_s, \rhd, q_1, \rhd, +1\rangle$

  2: $\langle q_1, 0, q_1, \mathrm{b}, +1\rangle$

  3: $\langle q_1, 1, q_1, \mathrm{b}, +1\rangle$

  4: $\langle q_1, \mathrm{b}, q_2, \mathrm{b}, -1\rangle$

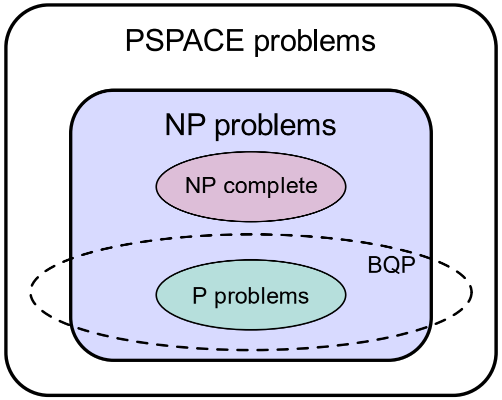
  5: $\langle q_2, \mathrm{b}, q_2, \mathrm{b}, -1\rangle$

  6: $\langle q_2, \rhd, q_3, \rhd, +1\rangle$

  7: $\langle q_3, \mathrm{b}, q_\mathrm{h}, 1{,}0\rangle$

Church-Turing thesis

- Church-Turing thesis
  - The class of functions computable by a Turing machine corresponds exactly to the class of functions which we would naturally regard as being computable by an algorithm
- Feasibility thesis
  - Complexity-theoretic (or extended) Church-Turing thesis
  - A probabilistic Turing machine can efficiently simulate any realistic model of computation, where “efficiently” means polynomial-time reduction
  - BPP (bounded-error probabilistic polynomial time) is the class of decision problems solvable by a probabilistic Turing machine in polynomial time with an error probability bounded away from 1/3 for all instances.
  - BQP (bounded-error quantum polynomial time) is the class of decision problems solvable by a quantum computer in polynomial time, with an error probability of at most 1/3 for all instances.
- Quantum complexity-theoretic Church-Turing thesis
  - A quantum Turing machine can efficiently simulate any realistic model of computation



The suspected relationship of BQP to other problem spaces (image from https://en.wikipedia.org/wiki/BQP)