CSE 127: Computer Security

Asymmetric Crypto, TLS, PKI and CT

Deian Stefan

Adopted slides from Kirill Levchenko and Dan Boneh

Asymmetric Cryptography

- Also called public key cryptography
- Two separate keys
  - Public key: known to everyone
  - Private key: used to decrypt and sign

Asymmetric Primitives

- Encryption and decryption
- Signing and verification

Asymmetric Keys

- Each user has a public and private key
- Keys related to each other in algorithm-dependent way
  - Need a key generation function
  - Keygen(r) = (pk, sk)
    - pk: public key
    - sk: secret key
    - r: random bits

Public-key encryption

- **Encryption:** (public key, plaintext) → ciphertext
  - $E_{pk}(m) = c$
- **Decryption:** (secret key, ciphertext) → plaintext
  - $D_{sk}(c) = m$

Encryption properties

- Encryption and decryption are inverse operations
  - $D_{sk}(E_{pk}(m)) = m$
- Secrecy: ciphertext reveals nothing about plaintext
  - Computationally hard to decrypt without secret key
- What's the point?
  - Anybody with your public key can send you a secret message!

Implementations

- ElGamal encryption (1985)
  - Based on Diffie-Helman key exchange (1976), itself invented by Diffie, Hellman, and Merkle
  - Computational basis: hardness of discrete logarithms
- RSA encryption (1978)
  - Invented by Rivest, Shamir, and Adleman
  - Computational basis: hardness of factoring

Digital signatures

- **Signing:** (secret key, message) → signature
  - $S_{sk}(m) = s$
- **Verification:** (public key, message, signature) → bool
  - $V_{pk}(m,s) = \text{true} \mid \text{false}$

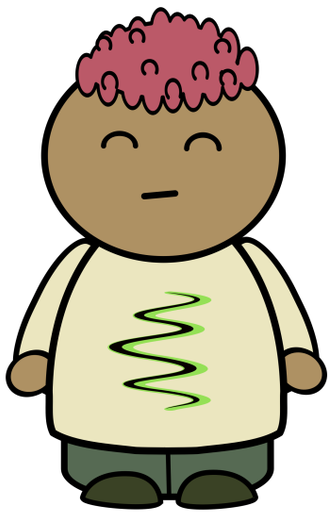
Signature properties

- Verification of signed message succeeds
  - $V_{pk}(m,S_{sk}(m))$ = true
- Unforgettability: can't compute signature for a message m without secret key sk
- What's the point?
  - Anybody with your public key can verify that you signed something!

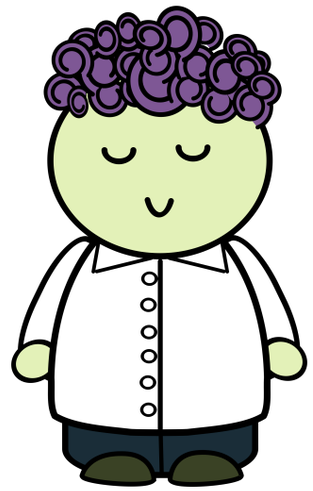
Implementations

- Digital Signature Algorithm (1991)
  - Closely related to ElGamal signature scheme (1984)
  - Computational basis: hardness of discrete logarithms
- RSA signatures
  - Invented by Rivest, Shamir, and Adleman
  - Computational basis: hardness of factoring

Encrypted and signed messaging



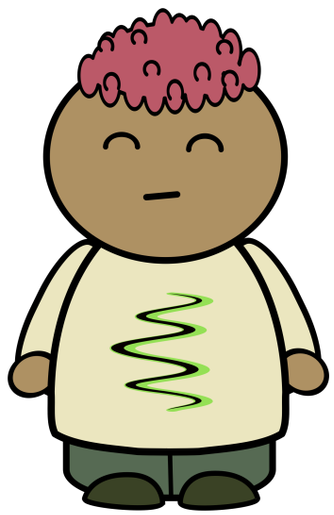
Alice



Bob

(this is a bad way to roll your own crypto)

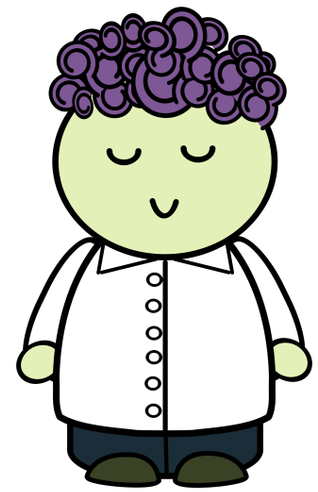
Encrypted and signed messaging



Alice

$(pk_{Alice\text{-}E}, sk_{Alice\text{-}E})$

$(pk_{Alice\text{-}S}, sk_{Alice\text{-}S})$



Bob

$(pk_{Bob\text{-}E}, sk_{Bob\text{-}E})$

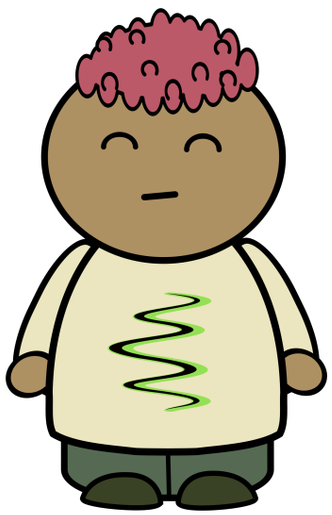
$(pk_{Bob\text{-}S}, sk_{Bob\text{-}S})$

(this is a bad way to roll your own crypto)

Encrypted and signed messaging

$E_{pkAlice\text{-}E}(m_1)=c_1$

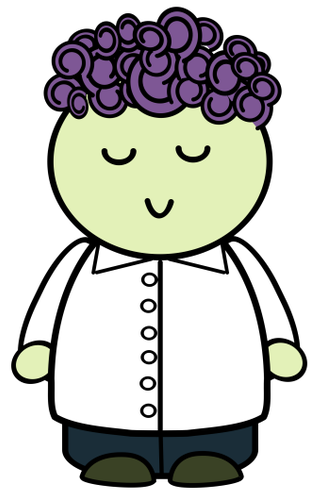
$(c_1, S_{skBob\text{-}S}(c_1))$



Alice

$(pk_{Alice\text{-}E}, sk_{Alice\text{-}E})$

$(pk_{Alice\text{-}S}, sk_{Alice\text{-}S})$



Bob

$(pk_{Bob\text{-}E}, sk_{Bob\text{-}E})$

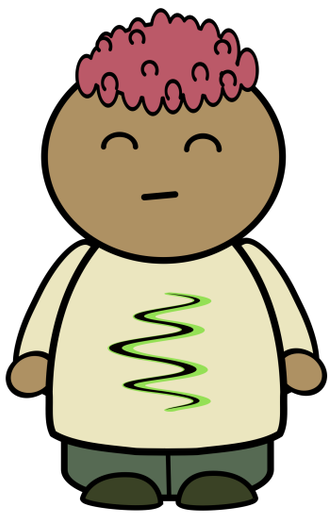
$(pk_{Bob\text{-}S}, sk_{Bob\text{-}S})$

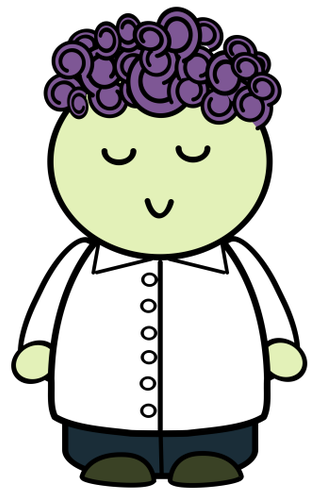
(this is a bad way to roll your own crypto)

Encrypted and signed messaging

$E_{pkAlice\text{-}E}(m_1)=c_1$

$(c_1, S_{skBob\text{-}S}(c_1))$





if $V_{pkBob\text{-}S}(c_1)$

$D_{skAlice\text{-}E}(c_1)$

Alice

$(pk_{Alice\text{-}E}, sk_{Alice\text{-}E})$

$(pk_{Alice\text{-}S}, sk_{Alice\text{-}S})$

Bob

$(pk_{Bob\text{-}E}, sk_{Bob\text{-}E})$

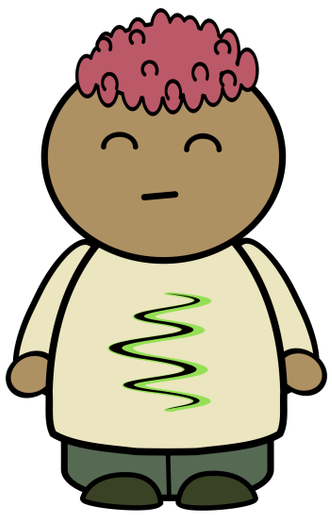
$(pk_{Bob\text{-}S}, sk_{Bob\text{-}S})$

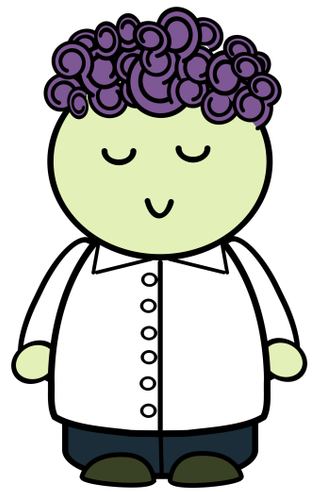
(this is a bad way to roll your own crypto)

Encrypted and signed messaging

$E_{pkAlice\text{-}E}(m_1)=c_1$

$(c_1, S_{skBob\text{-}S}(c_1))$





if $V_{pkBob\text{-}S}(c_1)$

$D_{skAlice\text{-}E}(c_1)$

Alice

$(pk_{Alice\text{-}E}, sk_{Alice\text{-}E})$

$(pk_{Alice\text{-}S}, sk_{Alice\text{-}S})$

Bob

$(pk_{Bob\text{-}E}, sk_{Bob\text{-}E})$

$(pk_{Bob\text{-}S}, sk_{Bob\text{-}S})$

(this is a bad way to roll your own crypto)

Practical Considerations

- Asymmetric cryptography operations are much more expensive than symmetric operations
  - Even implementations based on elliptic curves!
  - Don't want to encrypt/sign huge messages
- Moreover: asymmetric primitives operate on fixed-size messages

What do we do in practice?

- Usually combined with symmetric for performance
  - Use asymmetric to bootstrap ephemeral secret

Typical Encryption Usage

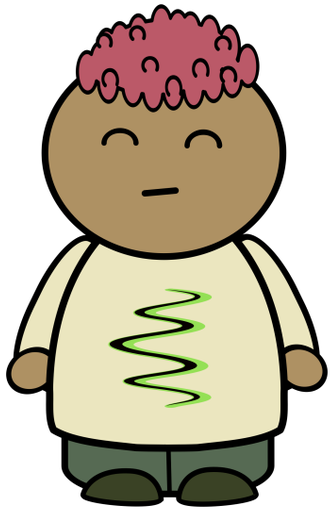
- Encryption:
  - Generate a ephemeral (one time) symmetric secret key
  - Encrypt message using ephemeral secret key
  - Encrypt ephemeral key using asymmetric encryption
- Decryption:
  - Decrypt ephemeral key, decrypt message

Typical Signature Usage

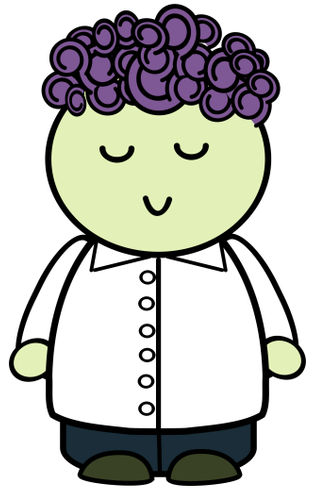
- Signing:
  - Compute cryptographic hash of message
  - Sign it using asymmetric signature scheme
- Verification:
  - Compute cryptographic hash of message
  - Verify it using asymmetric signature scheme

How do we get keys?

- Public keys are public: just ask for them!



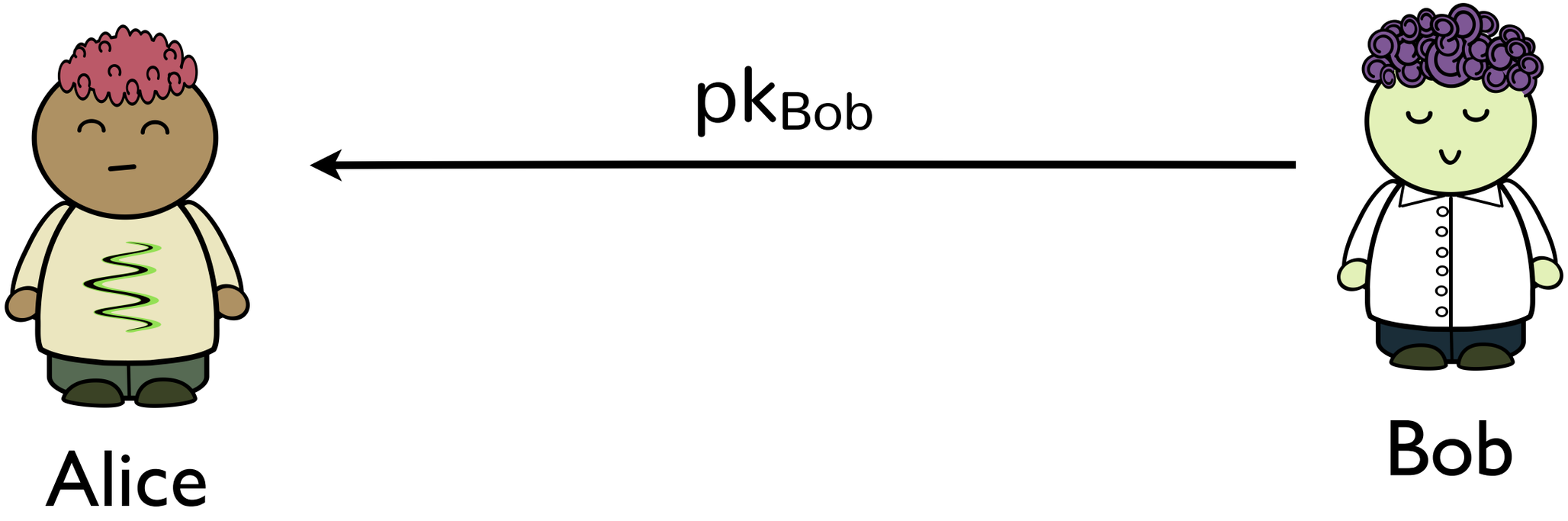
Alice



Bob

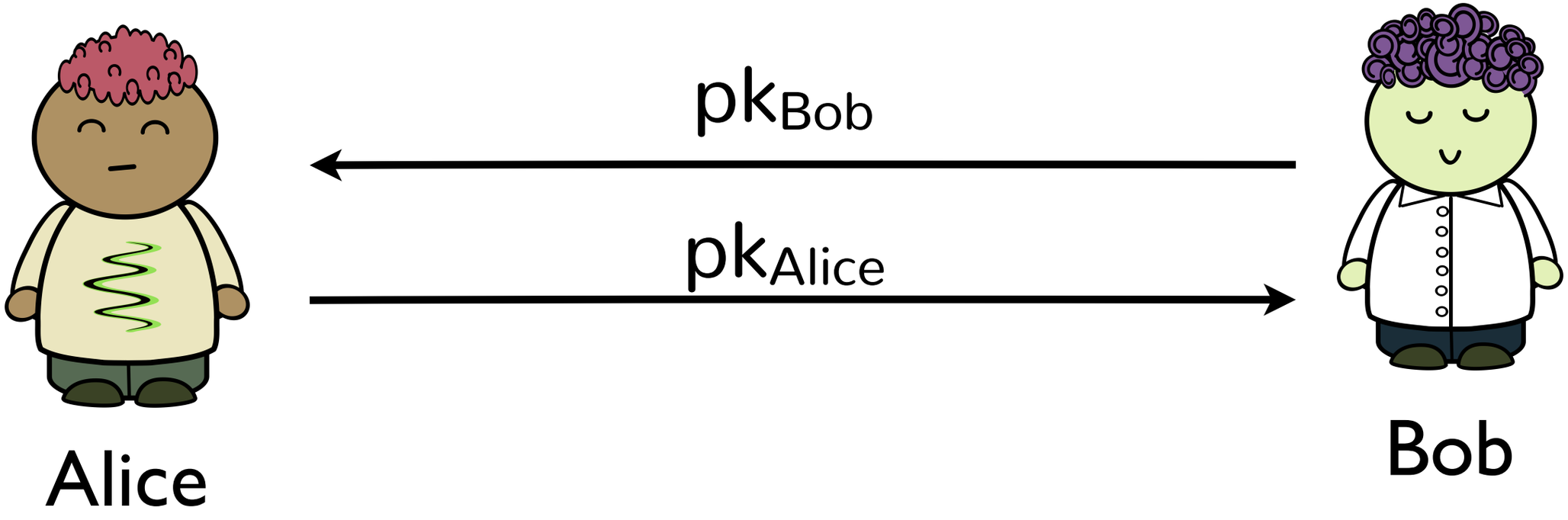
How do we get keys?

- Public keys are public: just ask for them!



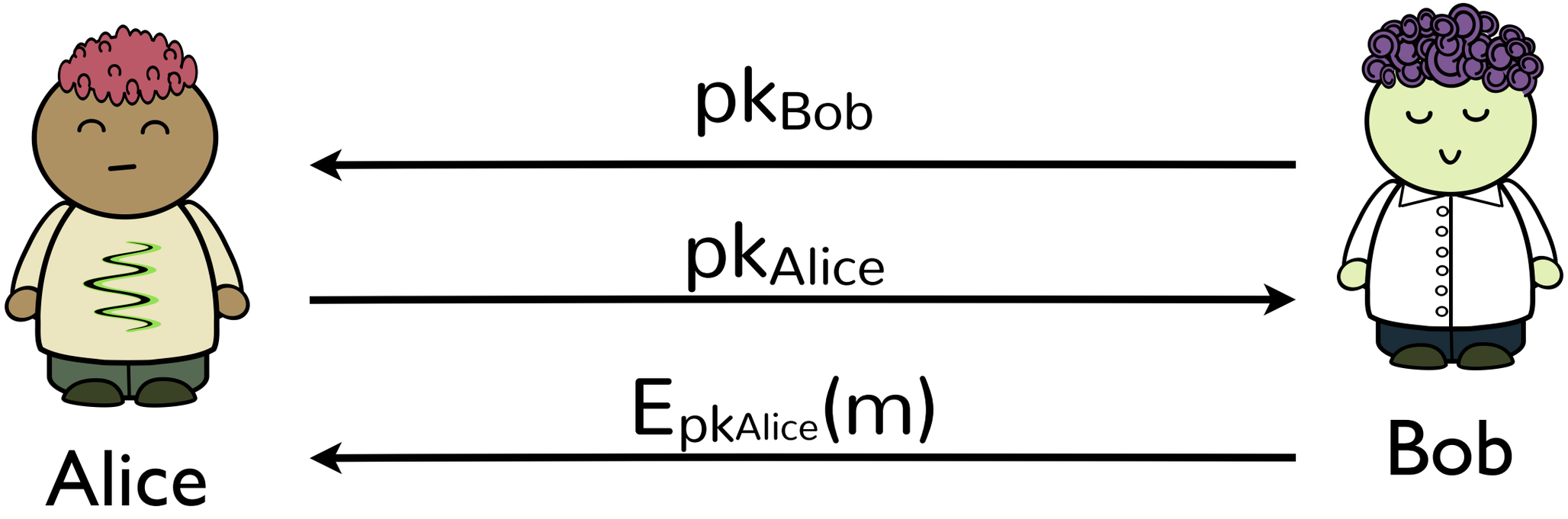
How do we get keys?

- Public keys are public: just ask for them!



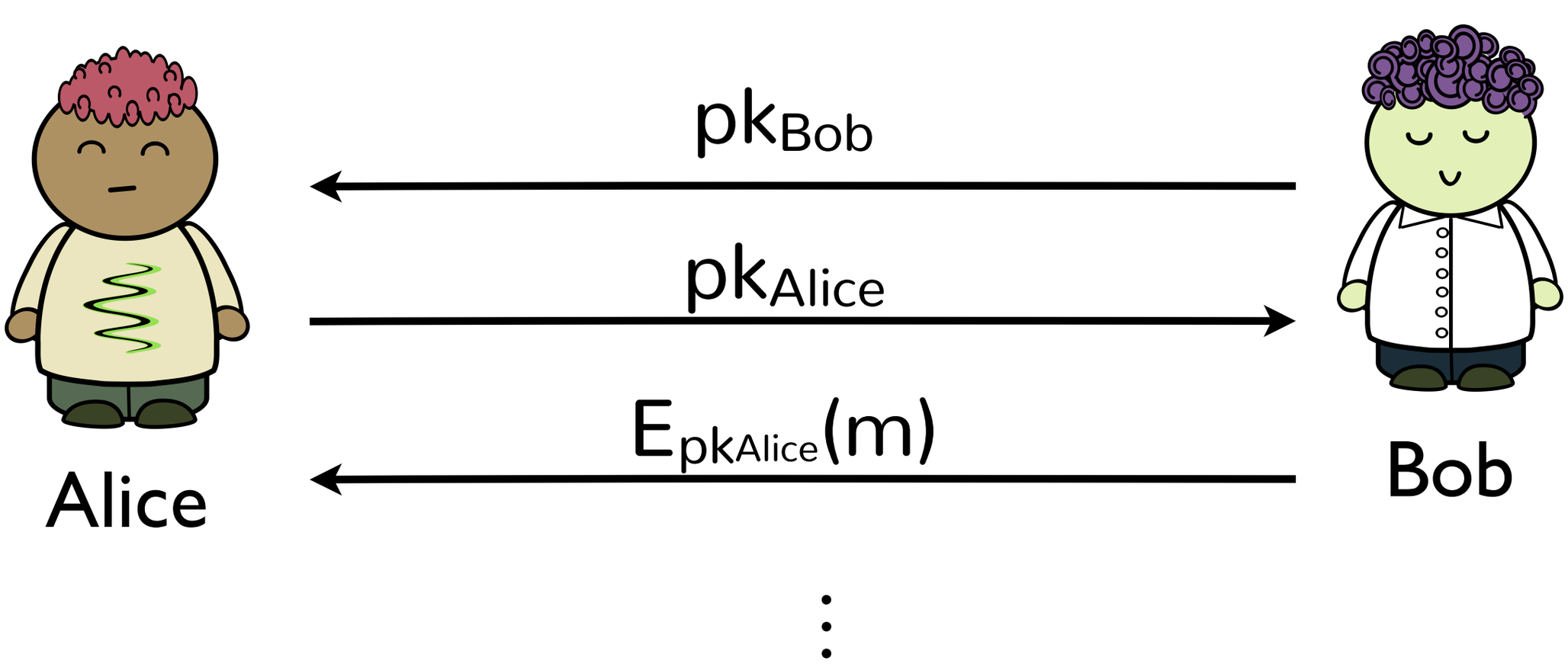
How do we get keys?

- Public keys are public: just ask for them!



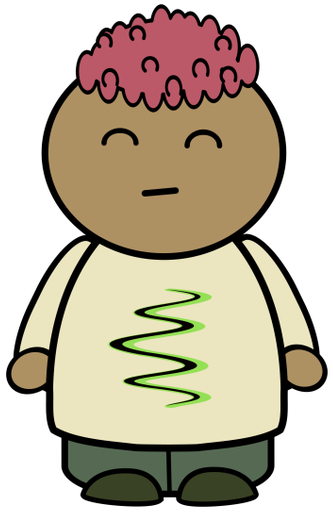
How do we get keys?

- Public keys are public: just ask for them!

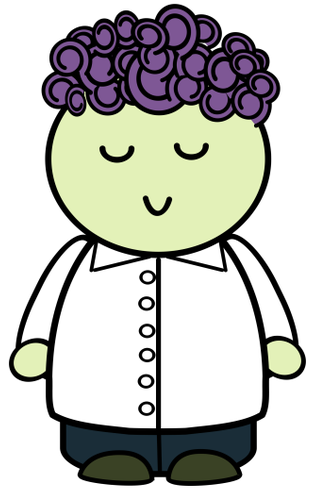


How do we get keys?

- Public keys are public: just ask for them!
  - No! Vulnerable to Man-in-the-Middle attacks!



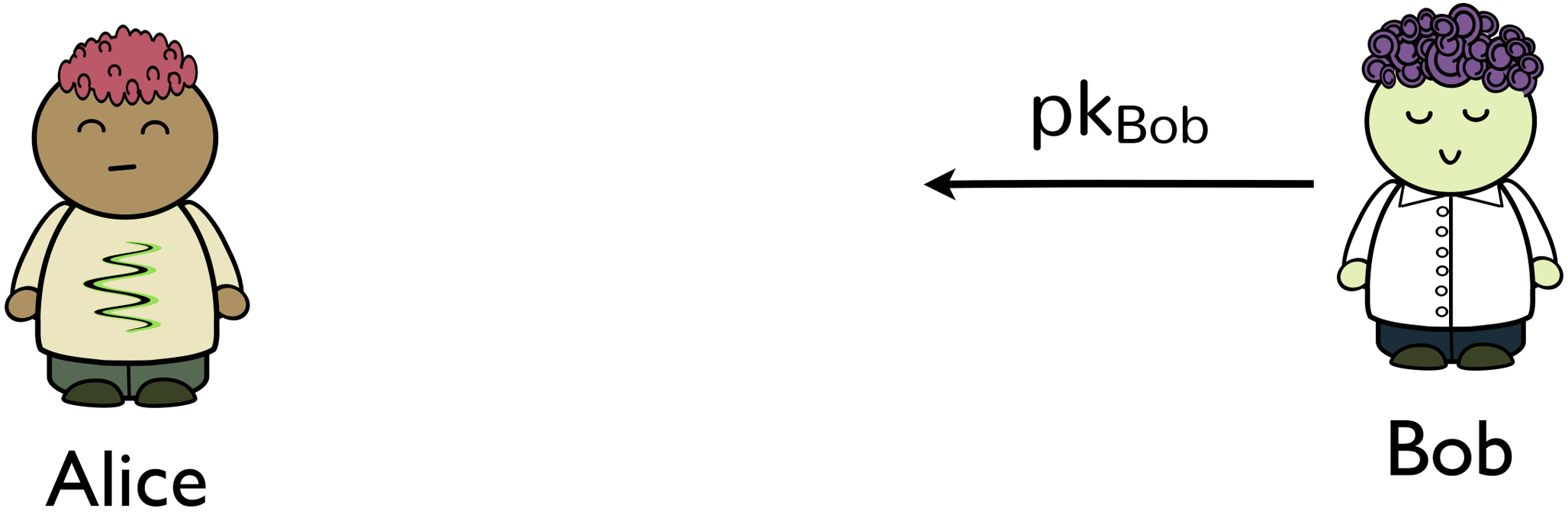
Alice



Bob

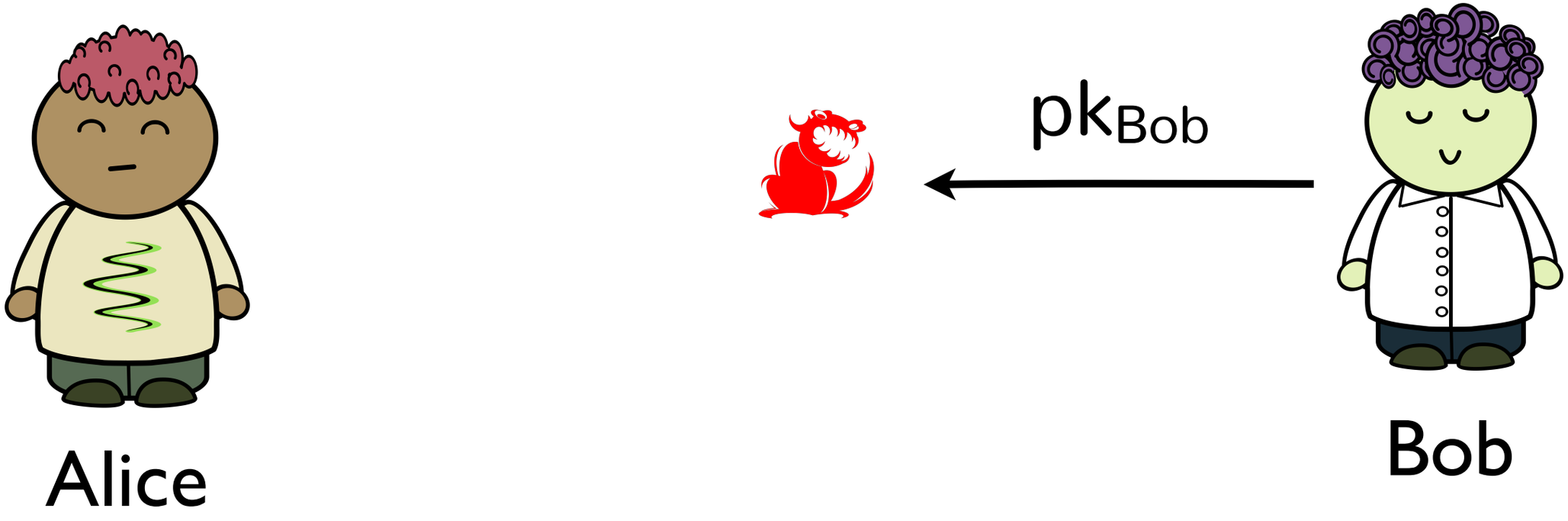
How do we get keys?

- Public keys are public: just ask for them!
  - No! Vulnerable to Man-in-the-Middle attacks!



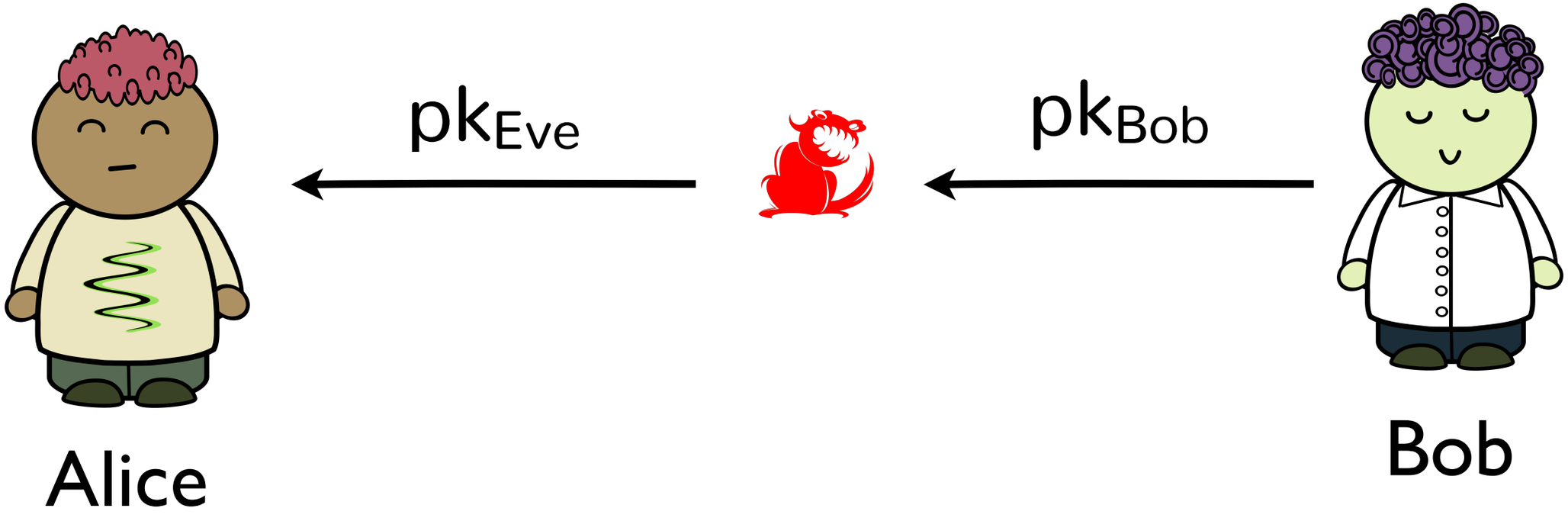
How do we get keys?

- Public keys are public: just ask for them!
  - No! Vulnerable to Man-in-the-Middle attacks!



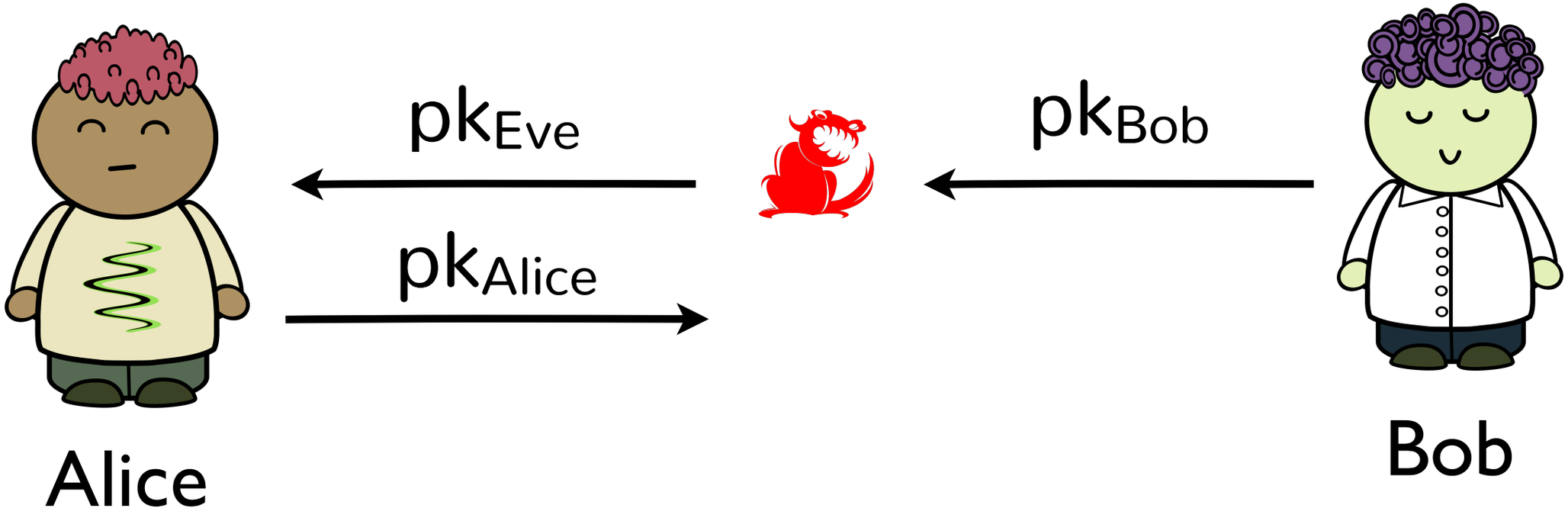
How do we get keys?

- Public keys are public: just ask for them!
  - No! Vulnerable to Man-in-the-Middle attacks!



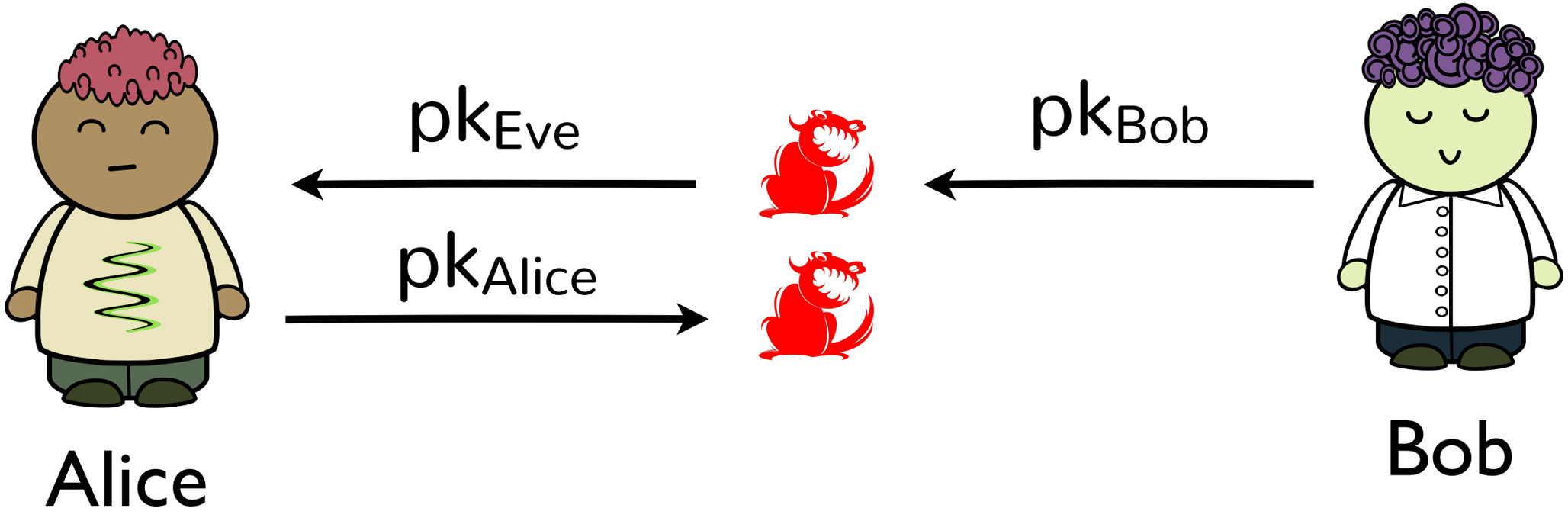
How do we get keys?

- Public keys are public: just ask for them!
  - No! Vulnerable to Man-in-the-Middle attacks!



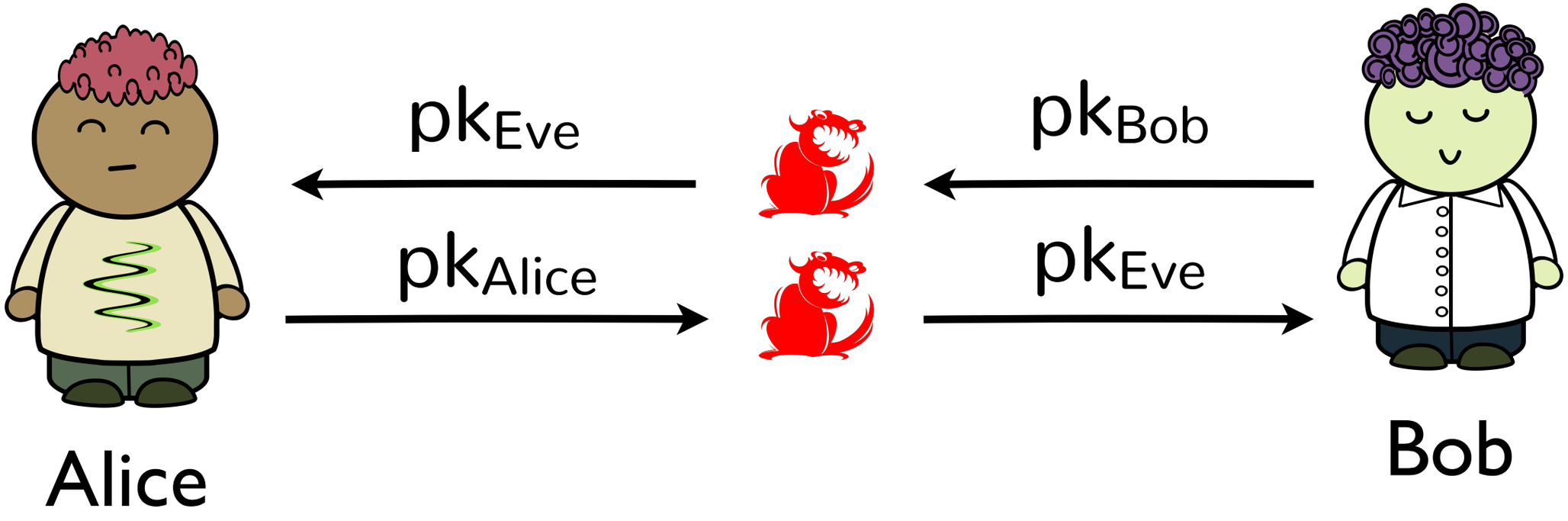
How do we get keys?

- Public keys are public: just ask for them!
  - No! Vulnerable to Man-in-the-Middle attacks!



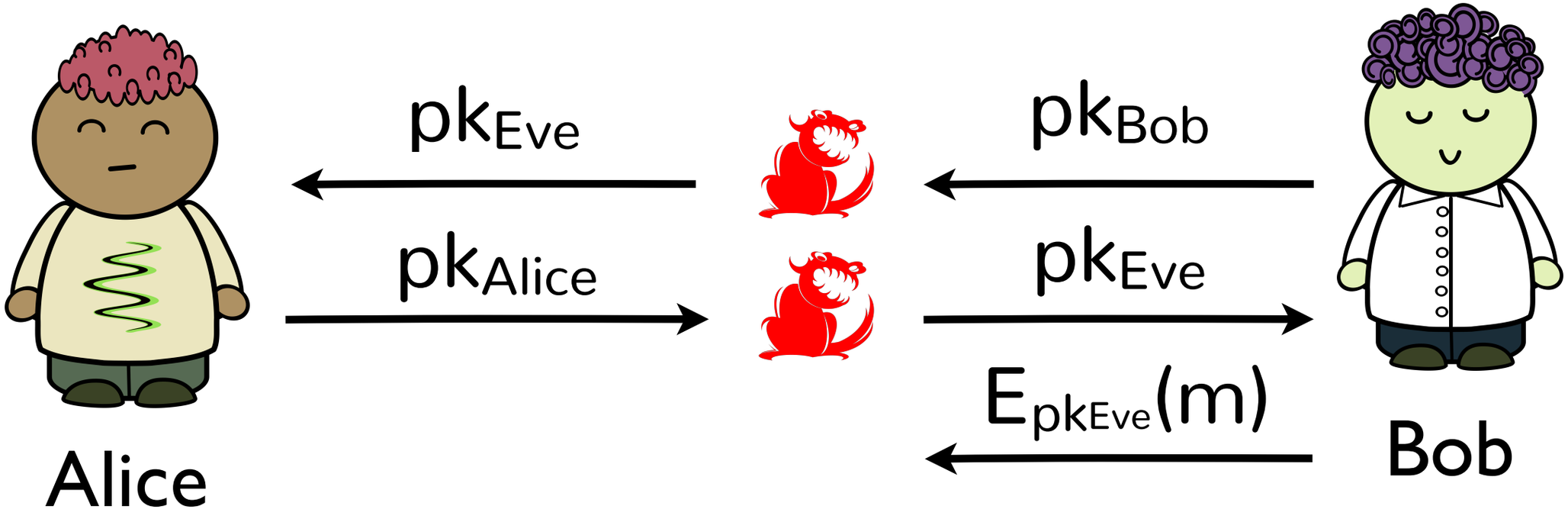
How do we get keys?

- Public keys are public: just ask for them!
  - No! Vulnerable to Man-in-the-Middle attacks!



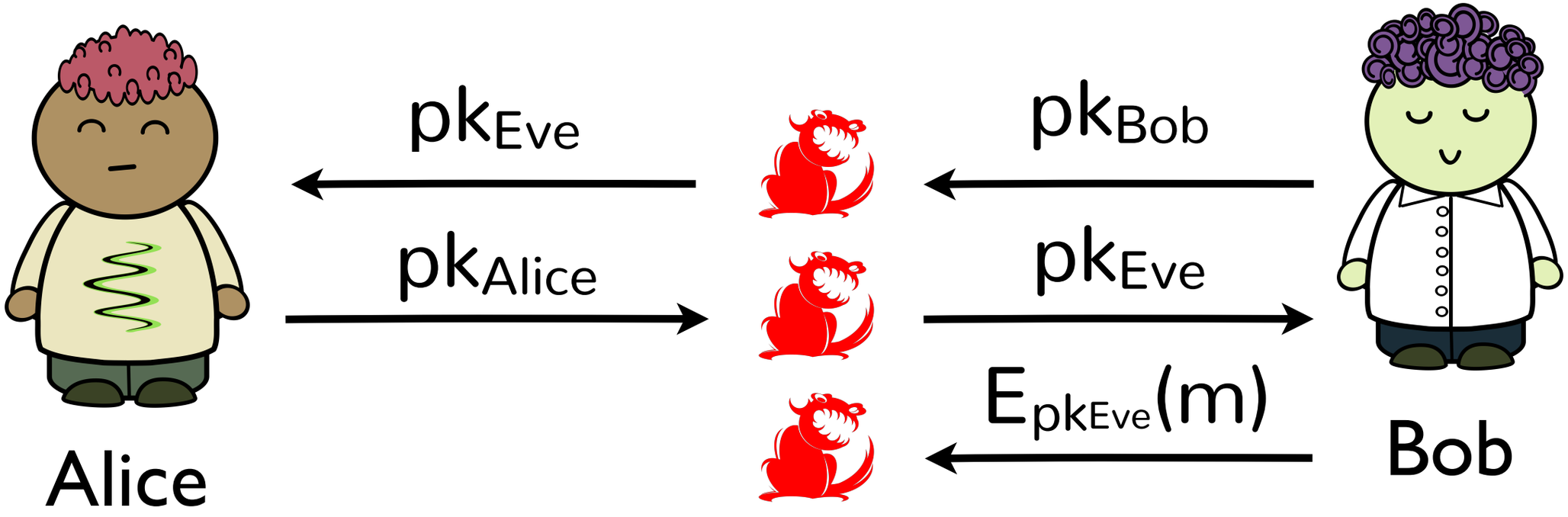
How do we get keys?

- Public keys are public: just ask for them!
  - ➤ No! Vulnerable to Man-in-the-Middle attacks!



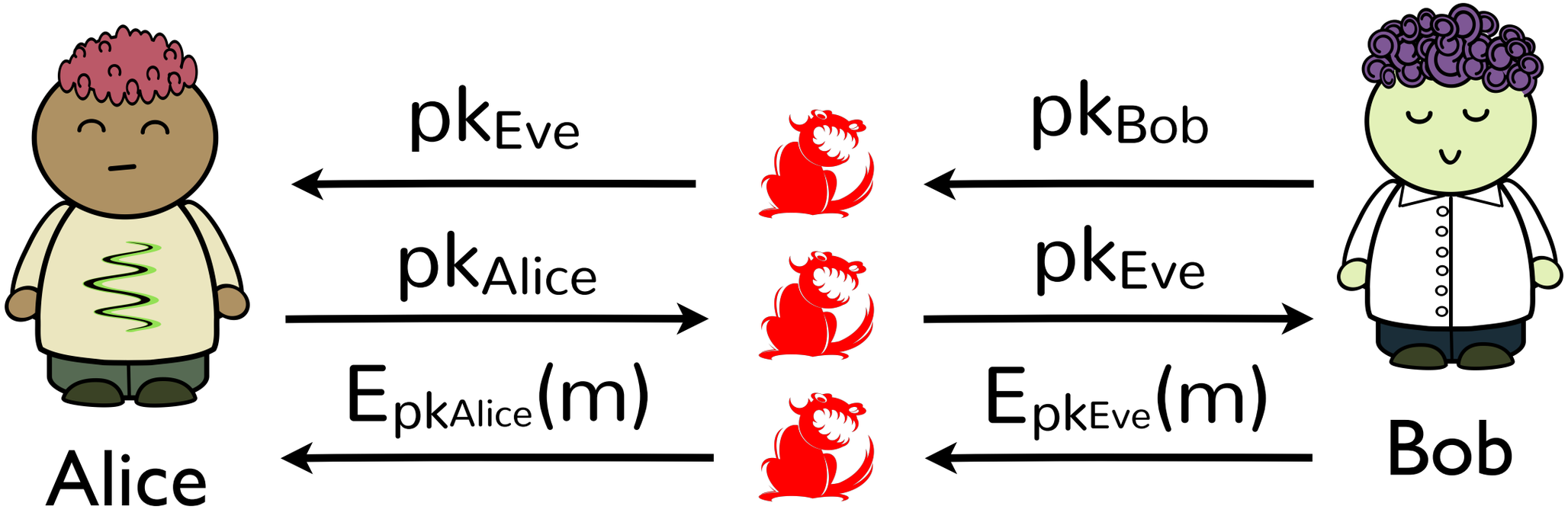
How do we get keys?

- Public keys are public: just ask for them!
  - ➤ No! Vulnerable to Man-in-the-Middle attacks!



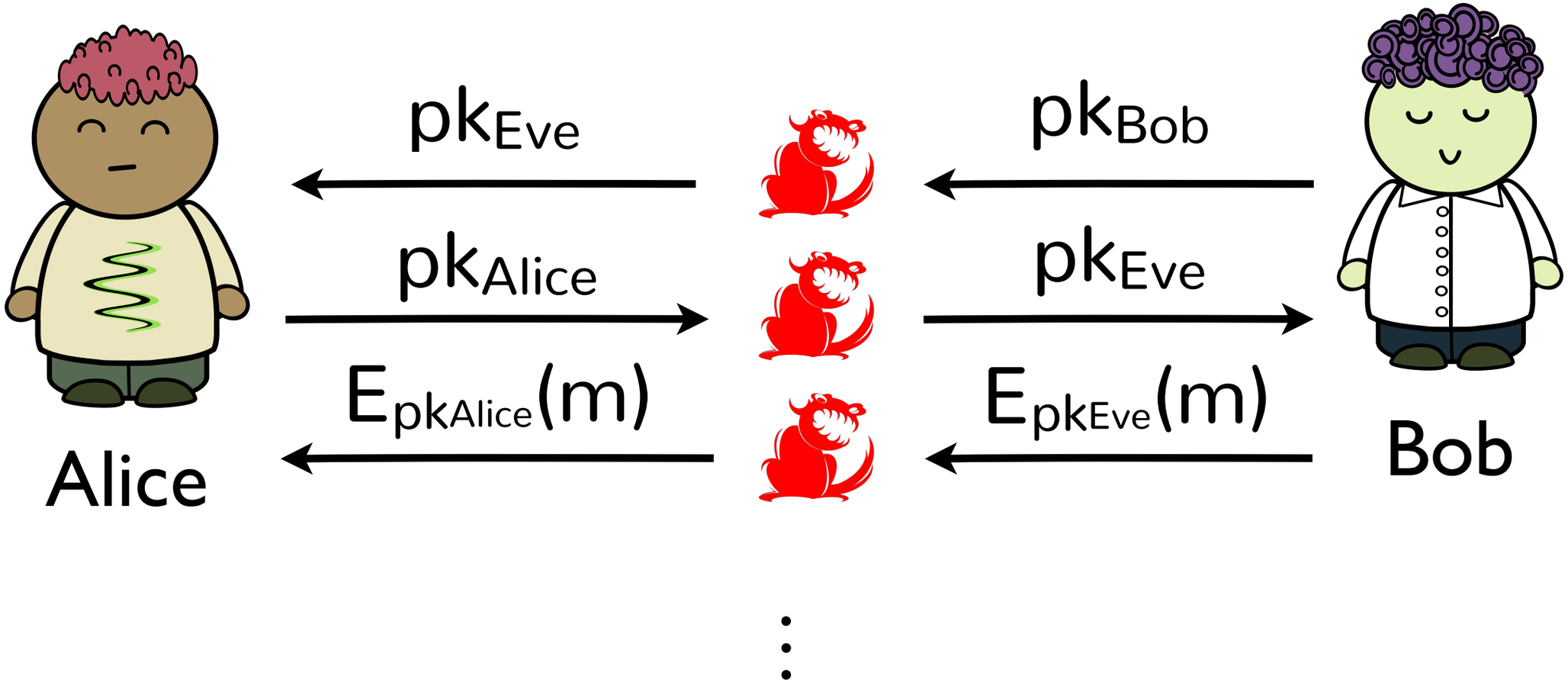
How do we get keys?

- Public keys are public: just ask for them!
  - No! Vulnerable to Man-in-the-Middle attacks!



How do we get keys?

- Public keys are public: just ask for them!
  - No! Vulnerable to Man-in-the-Middle attacks!



How do we get keys?

- Public directory contains everyone's public key
- To encrypt to a person, get their public key from directory
- No need for shared secrets!

Rest of slides soon...