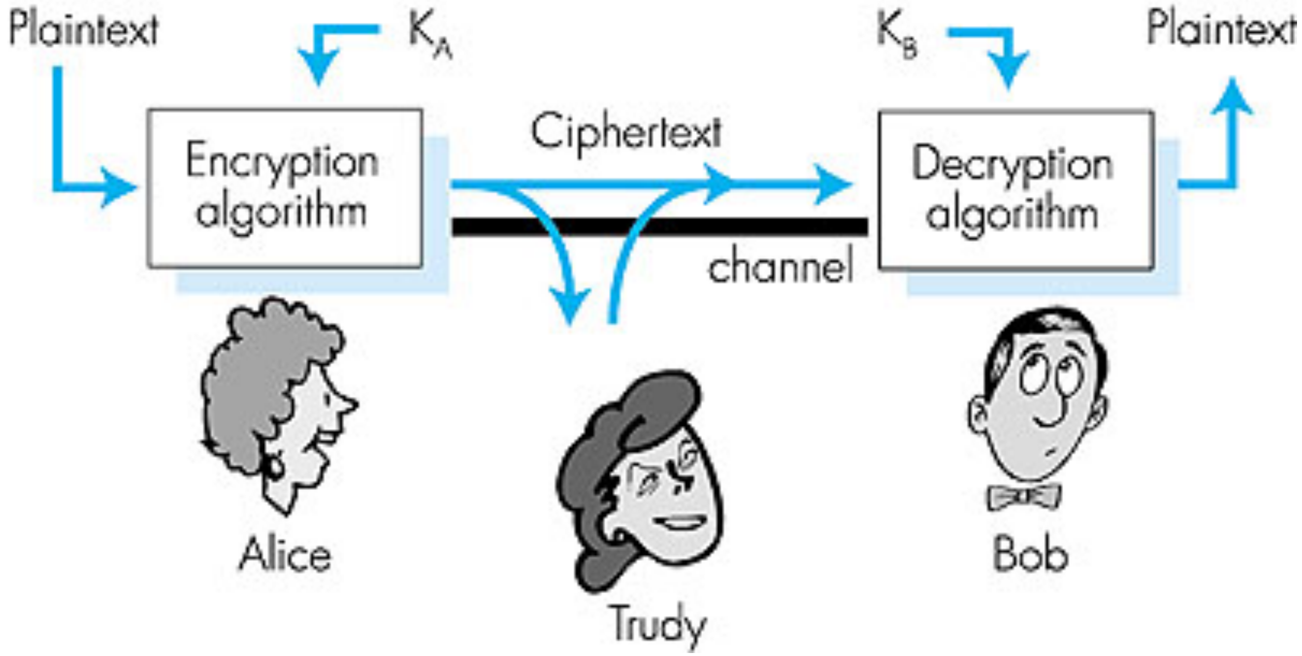
The language of cryptography

- Goal : secrecy, authentication and message integrity
- symmetric key crypto: sender, receiver keys identical
- public-key crypto: encrypt key *public*, decrypt key *secret*



Substitution cipher: substituting one thing for another

- *mono*alphabetic cipher: substitute *one* letter for another

```
plaintext:   abcdefghijklmnopqrstuvwxyz

Ciphertext:  mnbvcxzasdfghjklpoiuytrewq
```

E.g.:

```
Plaintext: bob. i love you. alice
ciphertext: nkn. s gktc wky. mgsbc
```

Q: How hard to break this simple cipher?:

- •brute force (how hard?)
- •other?

Public Key Cryptography

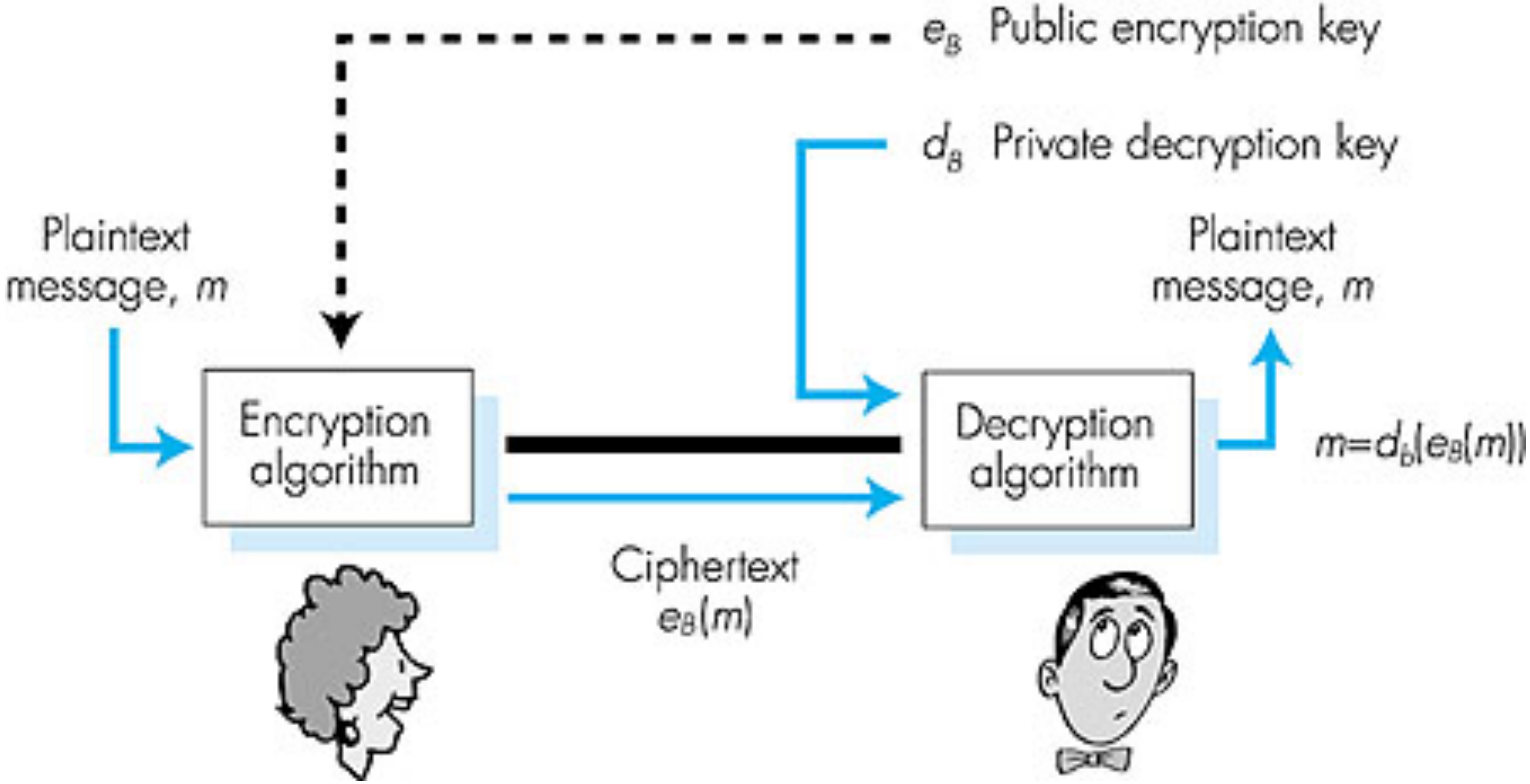
Symmetric key crypto

- requires sender, receiver know shared secret key
- Q: how to agree on key in first place (particularly if never "met")?

public key cryptography

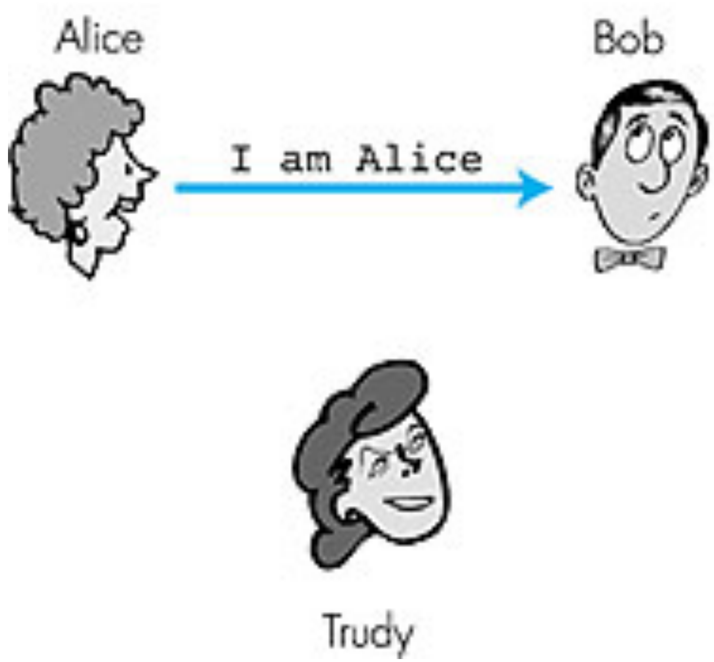
- radically different approach [Diffie-Hellman76, RSA78]
- sender, receiver do *not* share secret key
- encryption key *public* (known to *all*)
- decryption key private (known only to receiver)

Public Key Cryptography



Goal: Bob wants Alice to "prove" her identity to him

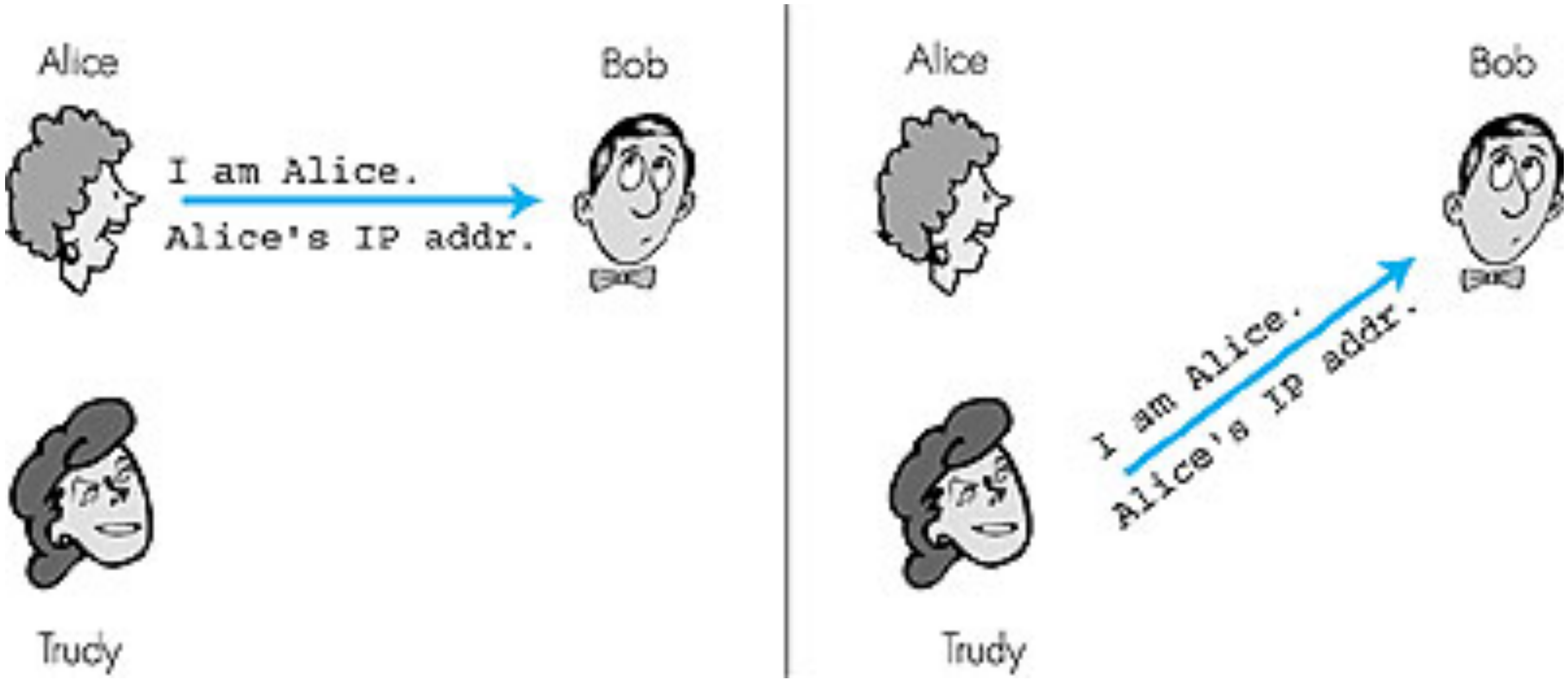
Protocol ap1.0: Alice says "I am Alice"



Failure scenario??

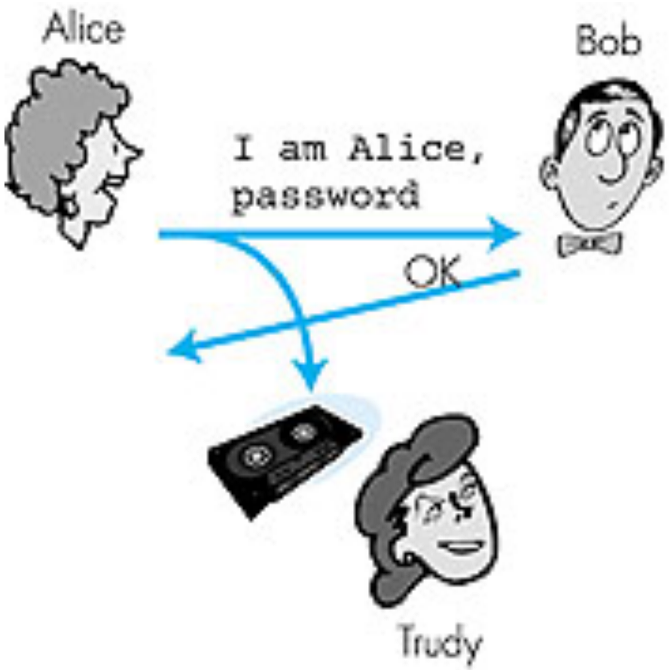
Authentication: another try

Protocol ap2.0: Alice says "I am Alice" and sends her IP address along to "prove" it.



Authentication: another try

Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.

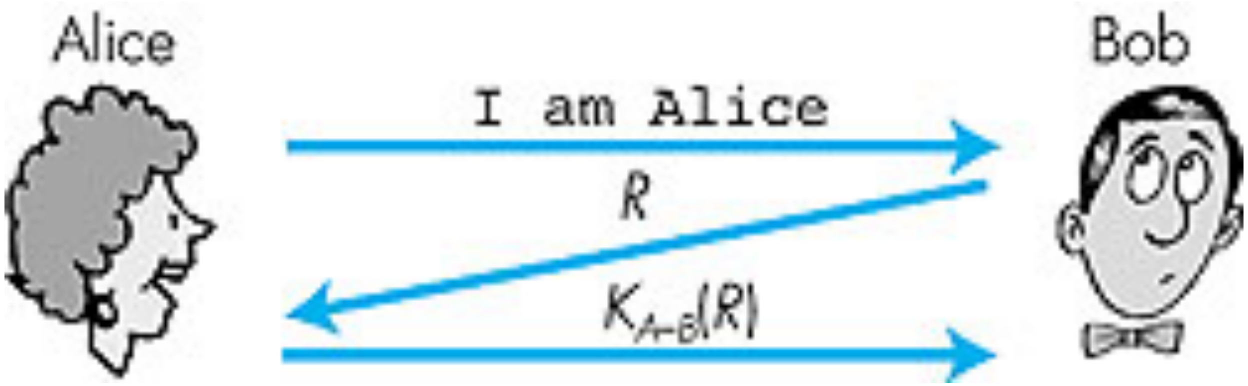


Failure scenario?

Authentication: yet another try

Goal: avoid playback attack

ap4.0: to prove Alice "live", Bob sends Alice nonce, R. Alice must return R, encrypted with shared secret key



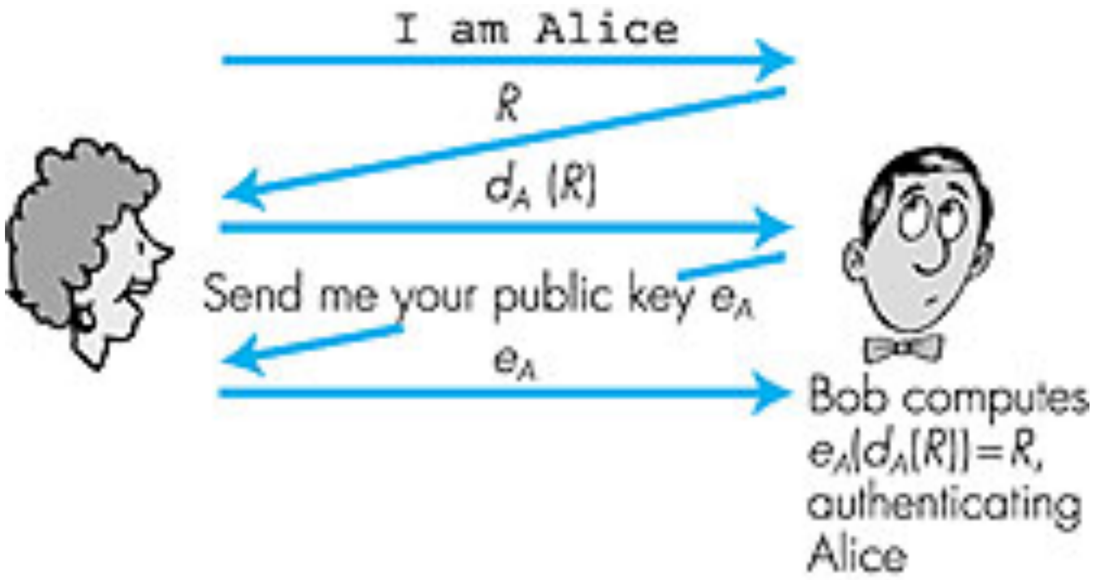
Failures, drawbacks?

Authentication: ap5.0

ap4.0 requires shared symmetric key

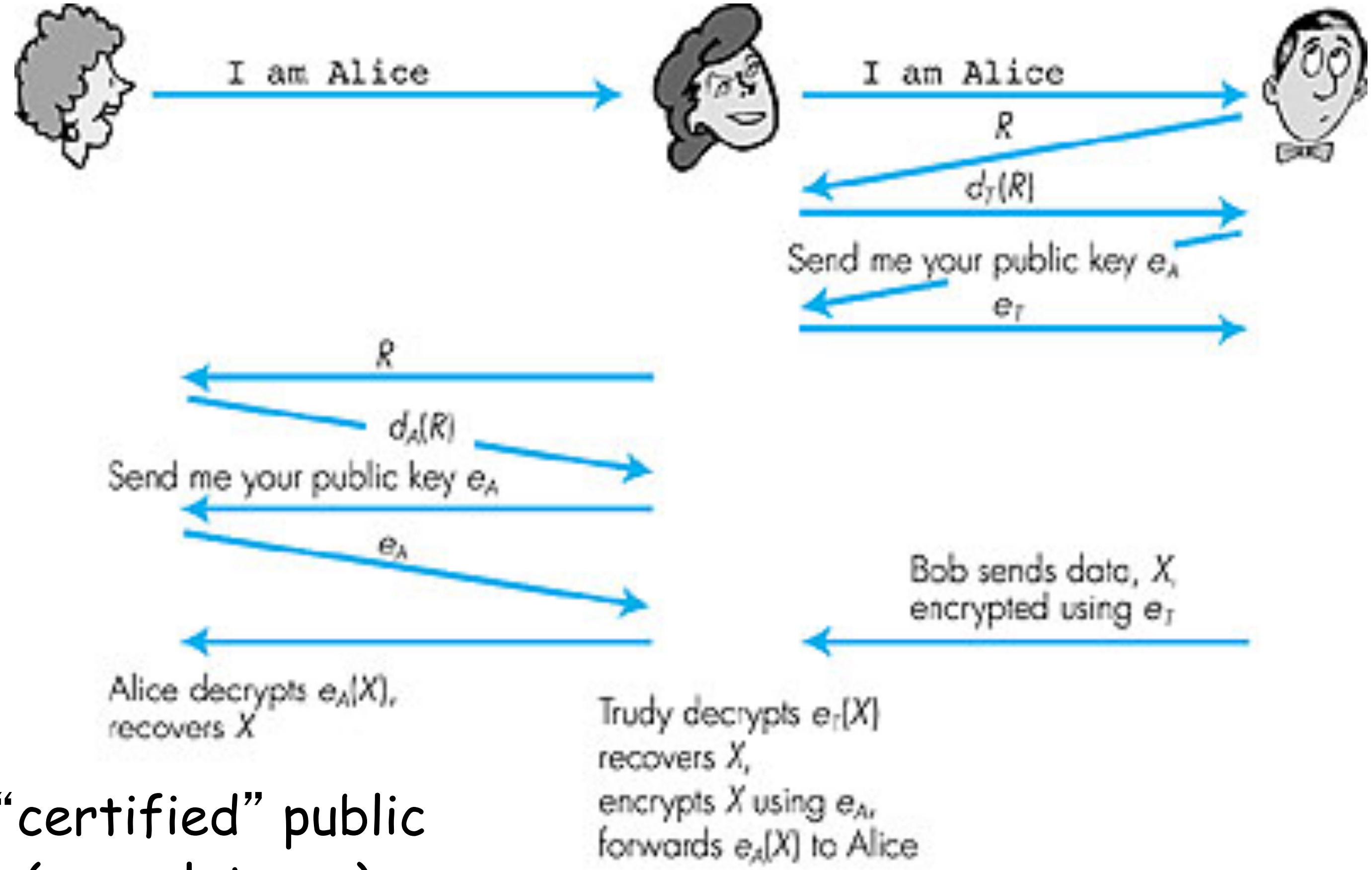
- problem: how do Bob, Alice agree on key
- can we authenticate using public key techniques?

ap5.0: use nonce, public key cryptography



ap5.0: security hole

Man (woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)



Need "certified" public keys (more later ...)

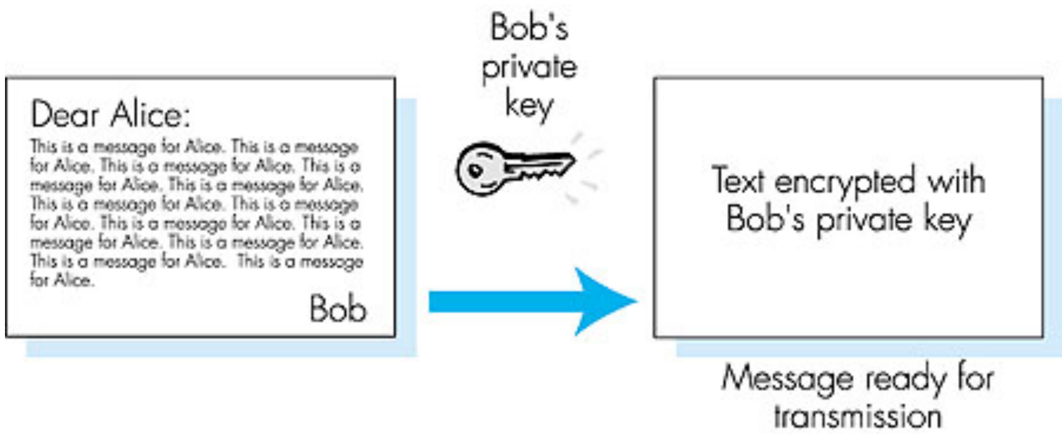
Digital Signatures

Cryptographic technique analogous to hand-written signatures.

- Sender (Bob) digitally signs document, establishing he is document owner/creator.
- Verifiable, nonforgeable: recipient (Alice) can verify that Bob, and no one else, signed document.

Simple digital signature for message m:

- Bob encrypts m with his private key d_B, creating signed message, $d_B(m)$.
- Bob sends m and $d_B(m)$ to Alice.



Digital Signatures (cont.)

- Suppose Alice receives msg *m*, and digital signature $d_B(m)$
- Alice verifies *m* signed by Bob by applying Bob's public key e_B to $d_B(m)$ *then* checks $e_B(d_B(m)) = m$.
- If $e_B(d_B(m)) = m$, whoever signed *m* must have used Bob's private key.

Alice thus verifies that:

- Bob signed *m*.
- No one else signed *m*.
- Bob signed m and not *m'*.

Non-repudiation:

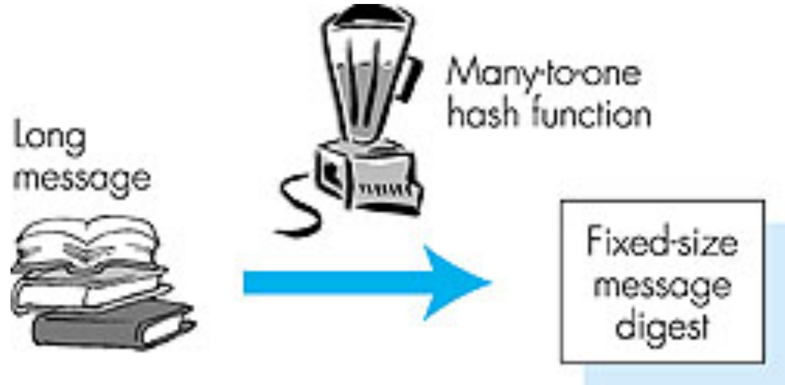
- Alice can take *m*, and signature $d_B(m)$ to court and prove that Bob signed *m*.

Message Digests

Computationally expensive to public-key-encrypt long messages

Goal: fixed-length,easy to compute digital signature, “fingerprint”

- **apply hash function H to *m*, get fixed size message digest, *H(m)*.**

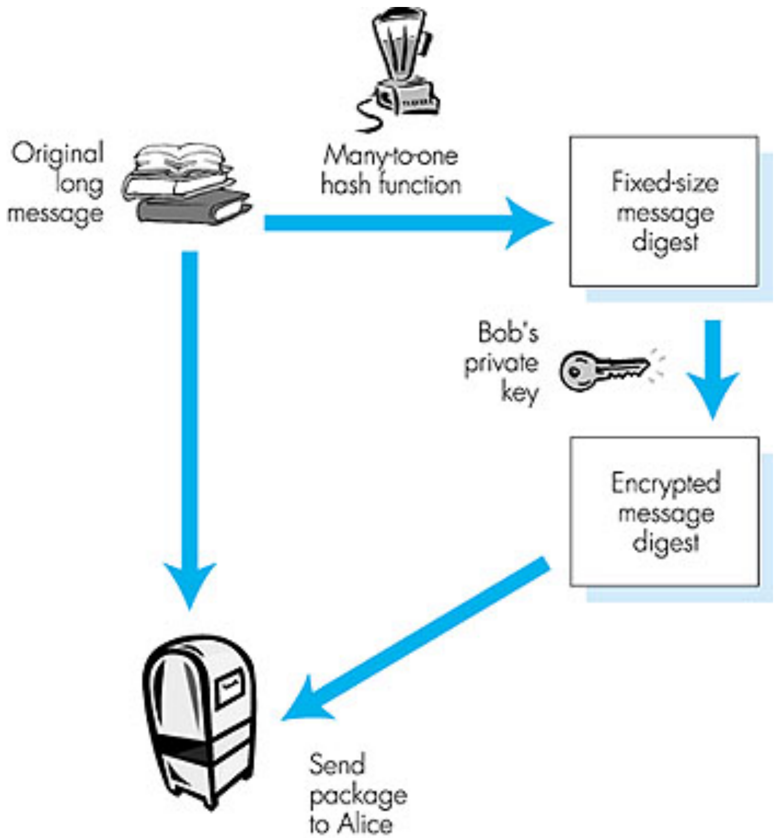


Hash function properties:

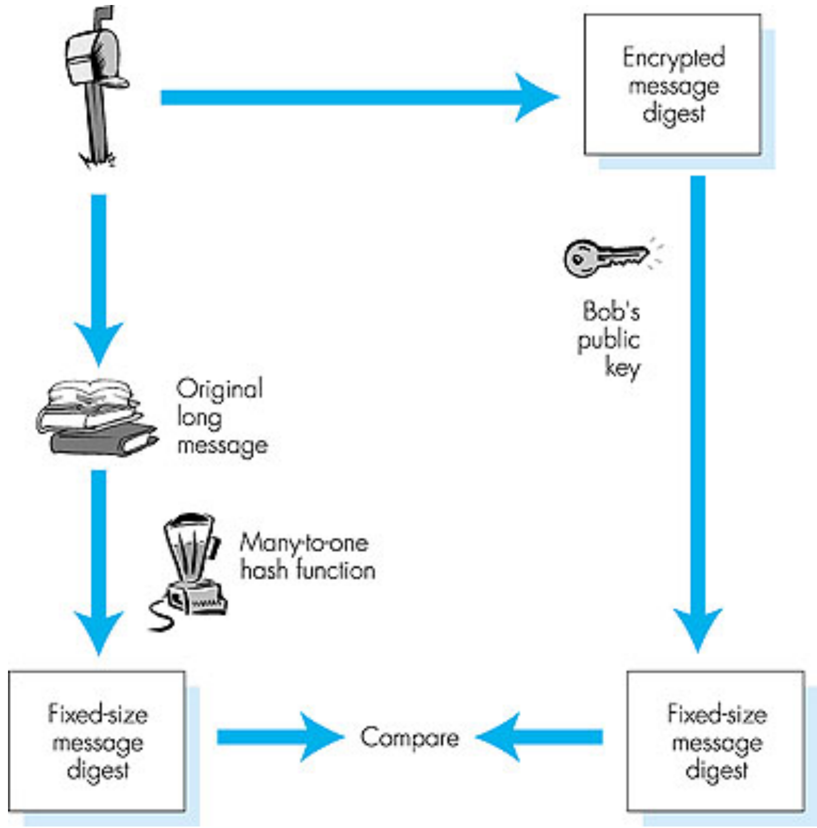
- Many-to-1
- Produces fixed-size msg digest (fingerprint)
- Given message digest x, computationally infeasible to find m such that x = H(m)
- computationally infeasible to find any two messages m and m’ such that H(m) = H(m’).

Digital signature = Signed message digest

Bob sends digitally signed message:

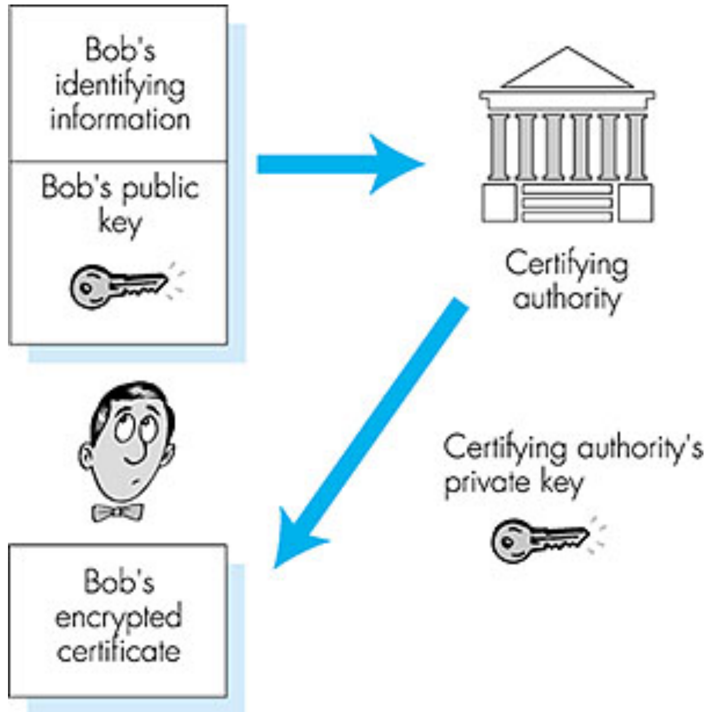


Alice verifies signature and integrity of digitally signed message:



Certificates

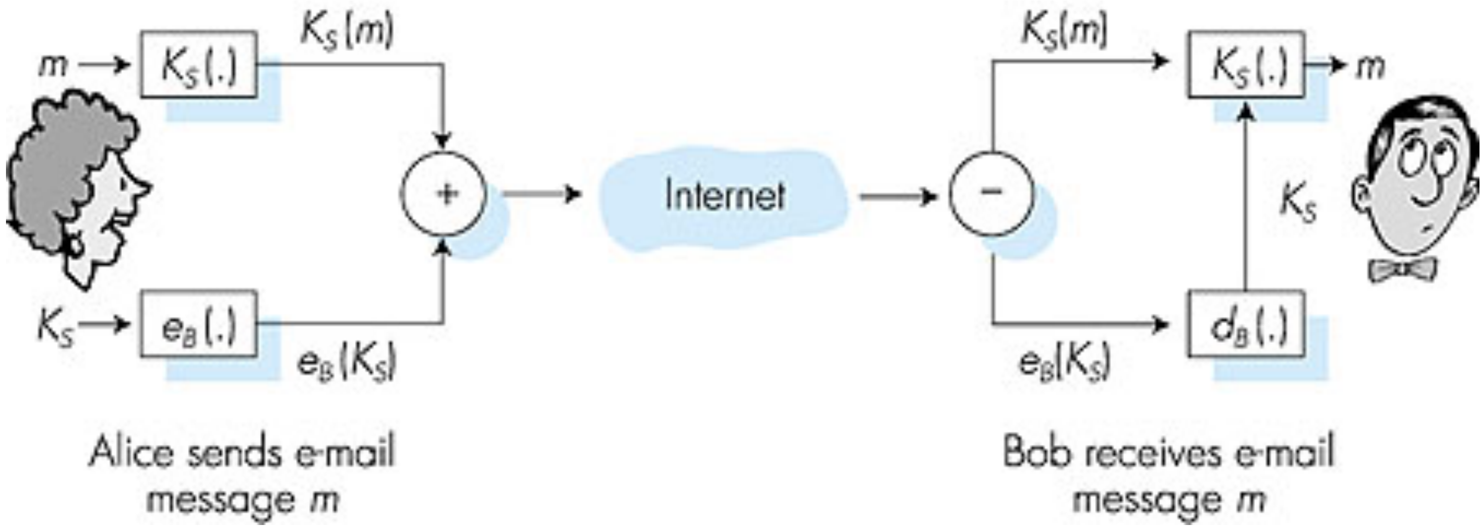
- **Certification authority (CA) binds public key to particular entity.**
- **Entity (person, router, etc.) can register its public key with CA.**
  - **Entity provides “proof of identity” to CA.**
  - **CA creates certificate binding entity to public key.**
  - **Certificate digitally signed by CA.**



- When Alice wants Bob’s public key:
- gets Bob’s certificate (Bob or elsewhere).
- Apply CA’s public key to Bob’s certificate, get Bob’s public key

Secure email

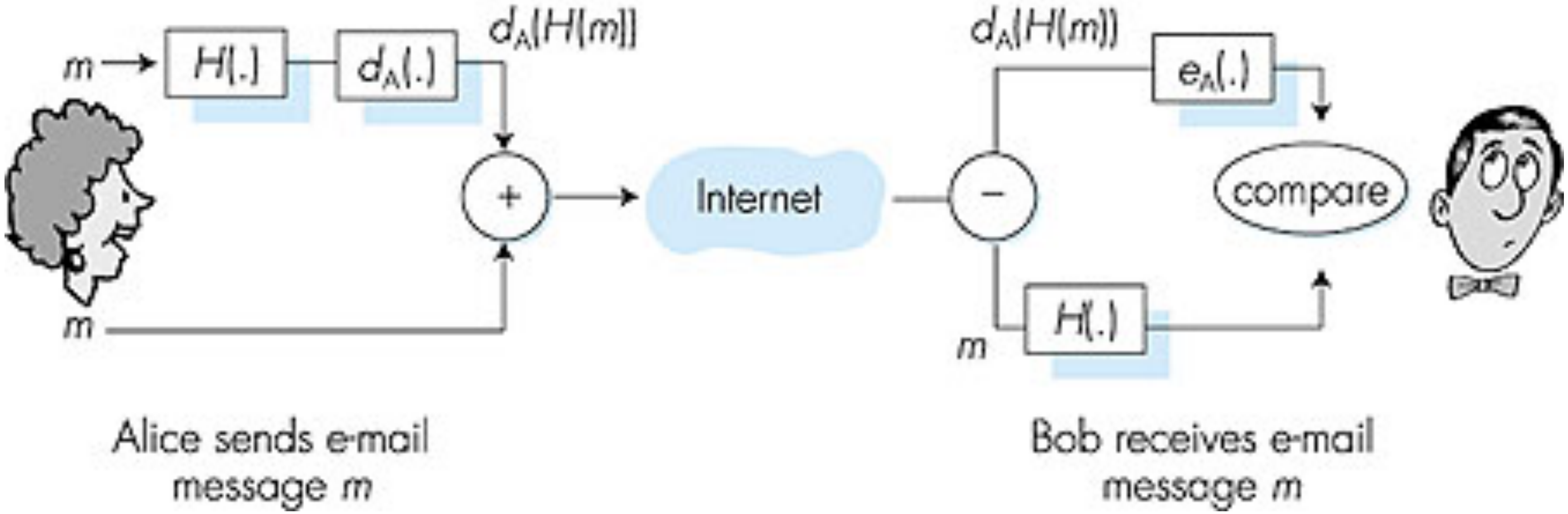
- Alice wants to send secret e-mail message, m, to Bob.



- generates random symmetric private key, K_S.
- encrypts message with K_S
- also encrypts K_S with Bob's public key.
- sends both $K_S(m)$ and $e_B(K_S)$ to Bob.

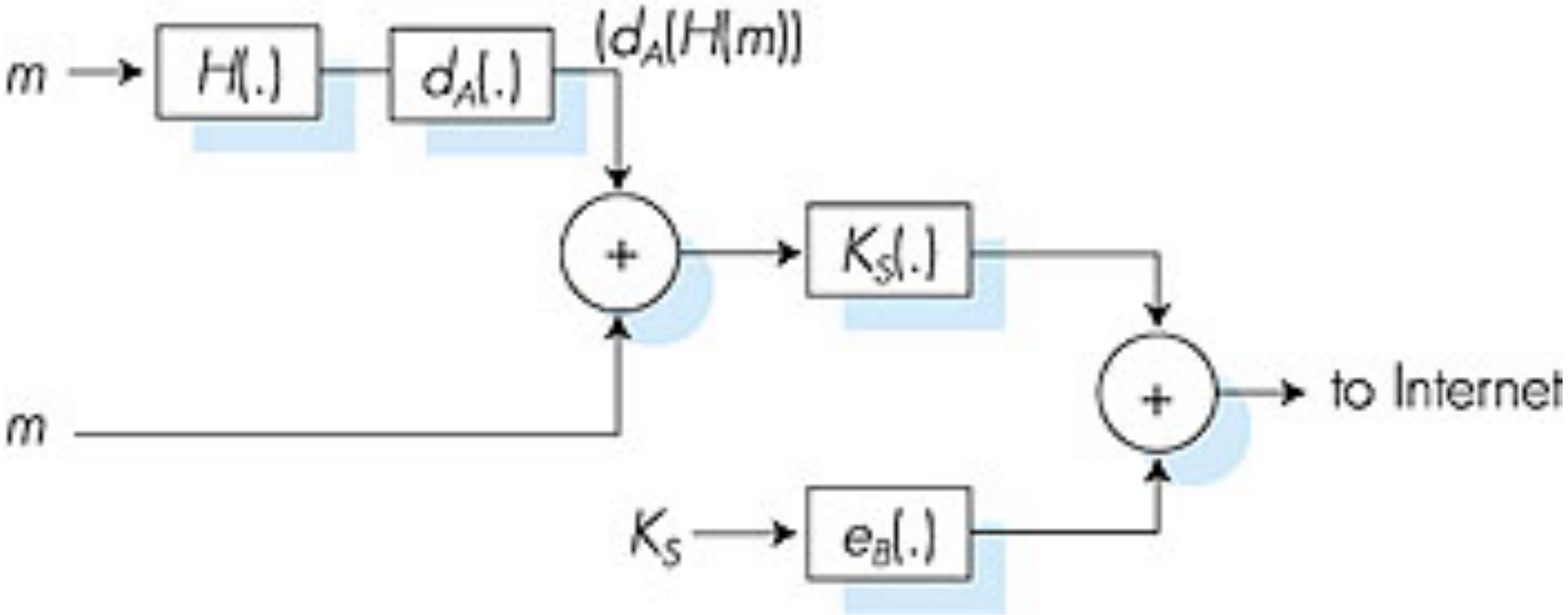
Email -adding authentication...

• Alice wants to provide sender authentication message integrity.



- Alice digitally signs message.
- sends both message (in the clear) and digital signature.

- Alice wants to provide secrecy, sender authentication, message integrity.



Note: Alice uses both her private key, Bob's public key.

This is what systems like PGP do. All secure email systems are slight variations of this scheme...