Collaborative Processing in Sensor Networ

Lecture 7 - Light-weight Security Solutions

Hairong Qi, Associate Professor
Electrical Engineering and Computer Science
University of Tennessee, Knoxville
http://www.eecs.utk.edu/faculty/qi
Email: hqi@utk.edu

Lecture Series at ZheJiang University, Summer 2008

Research Focus - Recap

- Develop energy-efficient collaborative processing algorithms with fault tolerance in sensor networks
  - Where to perform collaboration securely?
    - Computing paradigms
  - Who should participate in the collaboration securely?
    - Reactive clustering protocols
    - Sensor selection protocols
  - How to conduct collaboration securely?
    - In-network processing
    - Self deployment <--> Coverage

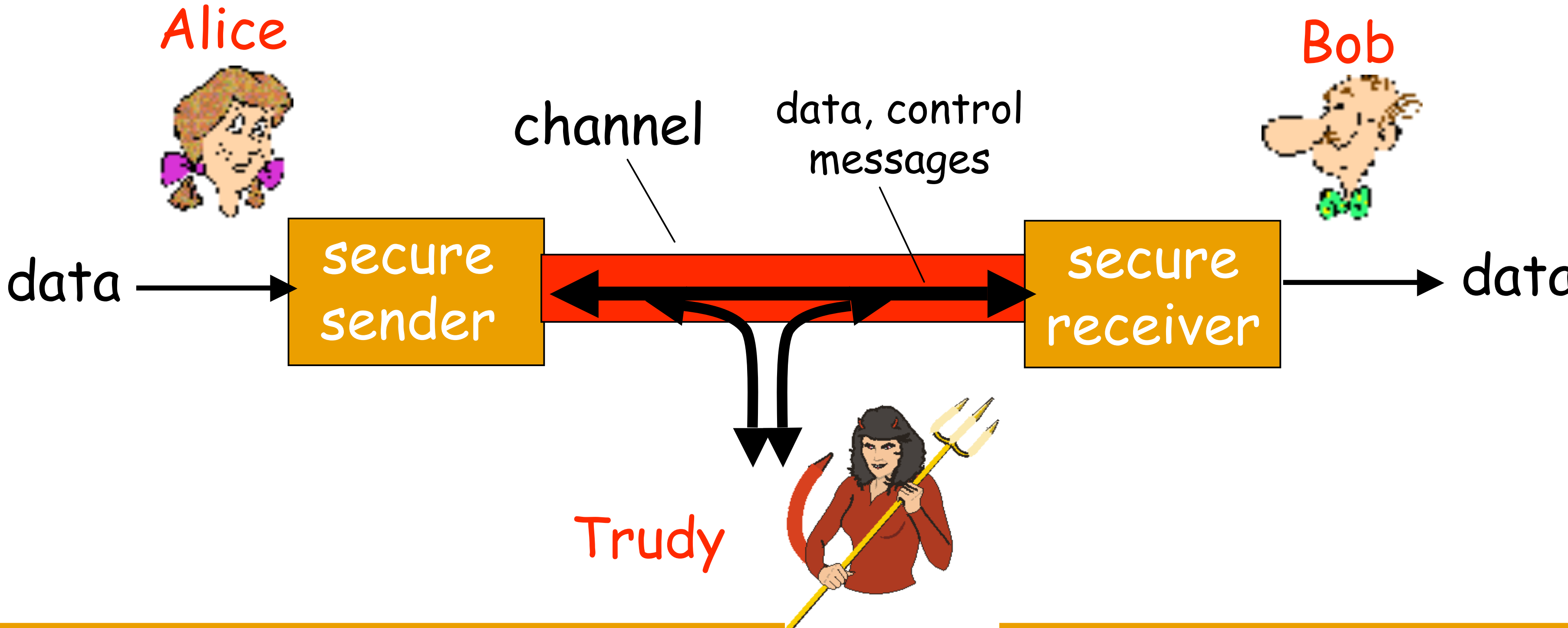
What is Network Security?

AICIP RESEAR

- Confidentiality: only sender, intended receiver should “understand” message contents
  - sender encrypts message
  - receiver decrypts message
- Authentication: sender, receiver want to confirm identity of each other
- Message Integrity: sender, receiver want to ensure message not altered (in transit, or afterwards)
- Non-repudiation
- Access Control and Availability: services must be accessible and available to legitimate users (no DoS attacks)

Friends and Foes: Alice, Bob, Trudy

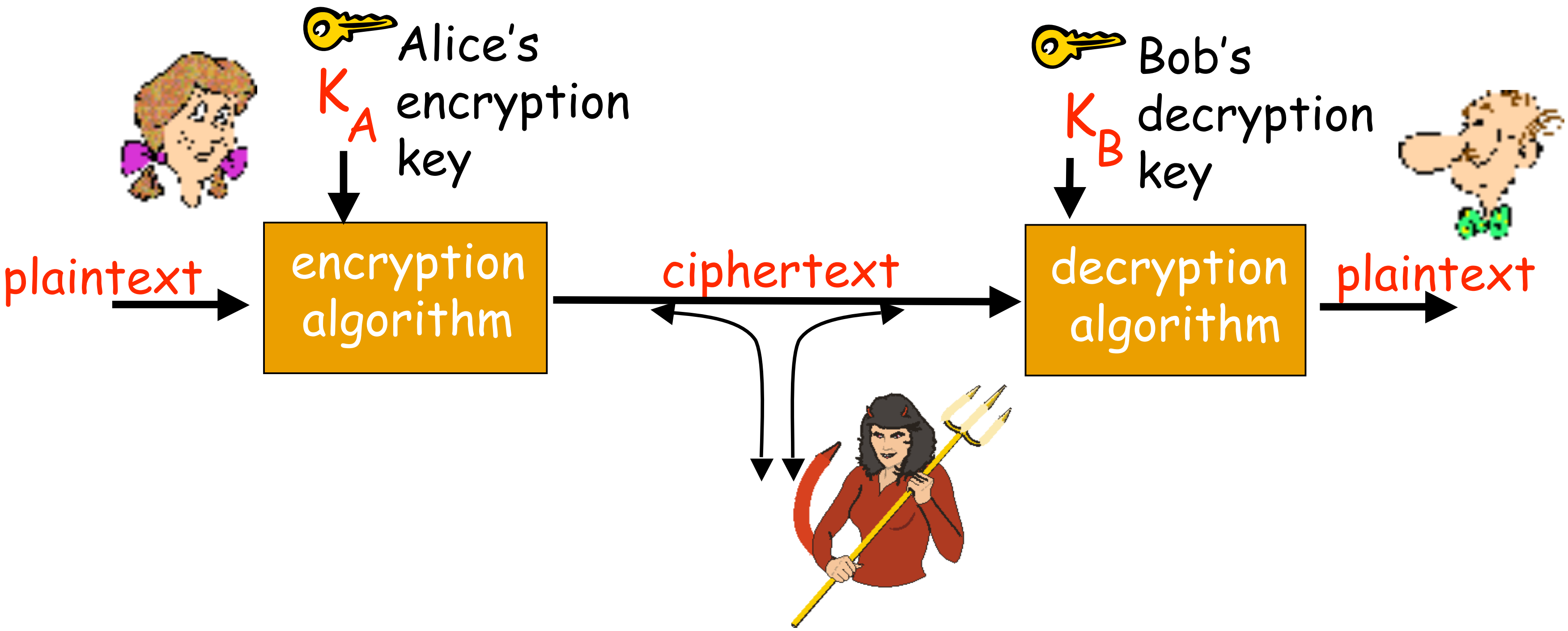
- Well-known fixtures in network security world
- Bob, Alice want to communicate "securely"
- Trudy (intruder) may intercept, delete, add message



What Can the “Enemy” Do ?

- A lot!
  - Eavesdrop: intercept messages
  - Actively insert messages into connection
  - Impersonation: can fake (spoof) source address in packet (or any field in packet)
  - Hijacking: “take over” ongoing connection by removing sender or receiver, inserting himself in place
  - Denial of service: prevent service from being used by others (e.g., by overloading resources)

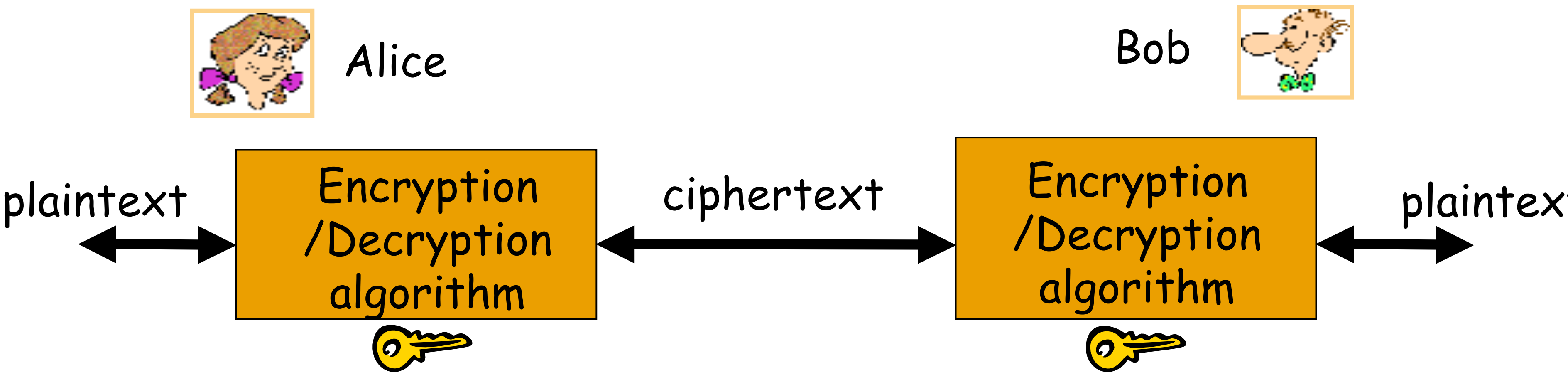
The Language of Cryptography



- Symmetric key crypto: sender, receiver keys identical
- Public-key crypto: encryption key public, decryption key secret (private)

Symmetric Key Cryptography

- All users (e.g., Bob and Alice) share and know the same (symmetric) key: K (e.g., DES)
- Encryption and decryption algorithms are identical



Problem: How can Bob and Alice share the same key in the first place?

Public Key Cryptography

- Radically different approach [Diffie-Hellman76, RSA78]
  - Uncovered an entire new approach to cryptography
  - W. Diffie and M.E. Hellman, “New Directions in Cryptography,” *IEEE transactions on Information Theory*, IT 22:644-654, 1976.
- Sender, receiver do not share secret key
- Public encryption key known to all
- Private decryption key known only to receiver

Diffie-Hellman Key Generation

(X - private key) A

a,p: known numbers
(p - prime number)

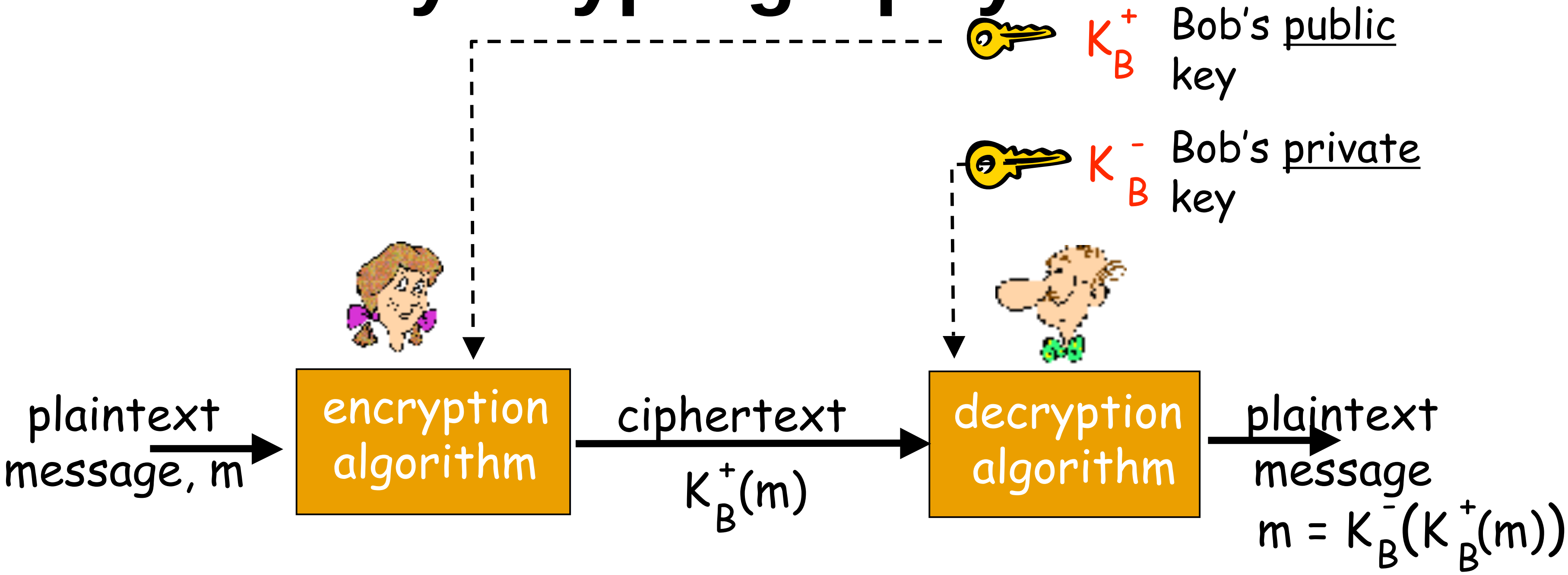
B (Y - private key)

$a^Y \bmod p$ $\quad$ $a^X \bmod p$

$$[a^Y \bmod p]^X \bmod p = \underline{a^{XY} \bmod p} = [a^X \bmod p]^Y \bmod p$$

- $x,y,a,p \rightarrow$ typically 1024 bits long
- The *Discrete Log* problem: by knowing $a^x \bmod p$, a and p, one <u>cannot</u> obtain x

Public Key Cryptography



Requirements:

1. $K_B^-(K_B^+(m)) = m$

2. Given a public key it should be impossible to compute the private key

RSA (Rivest-Shamir-Adelman): Choosing Keys

AICIP RESEAR

1. Choose two large prime numbers p, q. (e.g., 1024 bits each)

2. Compute $n = pq$, $z = (p-1)(q-1)$

3. Choose e (with $e<n$) that has no common factors with z. (e, z are "relatively prime").

4. Choose d such that $ed-1$ is exactly divisible by z. (in other words: $ed \bmod z = 1$).

5. *Public* key is $\underbrace{(n,e)}_{K_B^+}$. *Private* key is $\underbrace{(n,d)}_{K_B^-}$.

RSA: Encryption, Decryption

Given (n,e) and (n,d) as computed above:

1. To encrypt bit pattern, m $(m<n)$, compute

   $c = m^e \bmod n$ (i.e., remainder when m^e is divided by n)

2. To decrypt received bit pattern, c, compute

   $m = c^d \bmod n$ (i.e., remainder when c^d is divided by n)

Magic happens!

$$m = (\underbrace{m^e \bmod n}_{c})^d \bmod n$$

RSA: Why is That?

AICIP RESEAR

Useful number theory result: If p,q prime and $n = pq$, then:

$$x^y \bmod n = x^{y \bmod (p-1)(q-1)} \bmod n$$

(Fermat's Little Theorem)

$$\underbrace{(m^e \bmod n)}_{C}{}^d \bmod n = m^{ed} \bmod n$$

C - the encrypted message

$$= m^{ed \bmod (p-1)(q-1)} \bmod n$$

(using number theory result above)

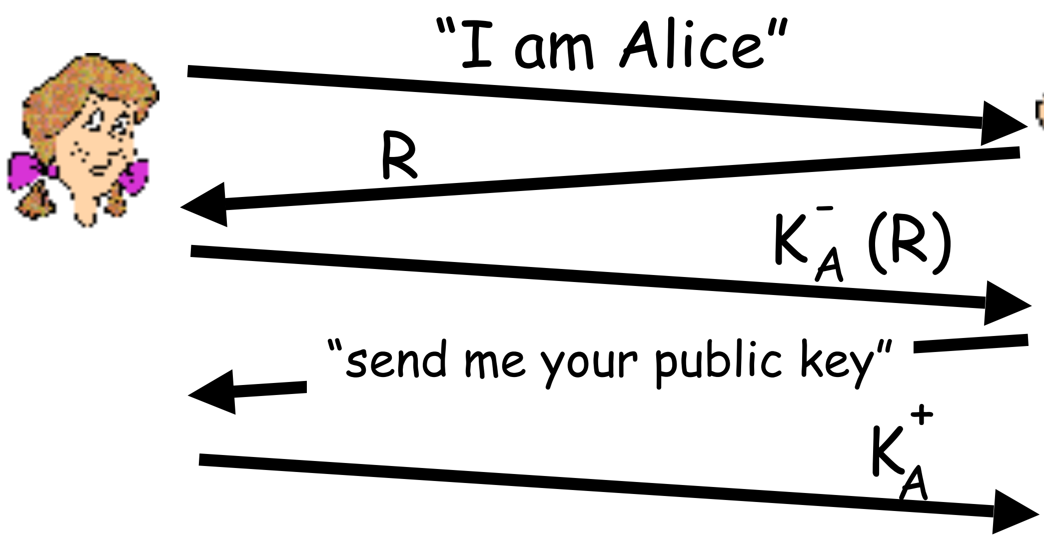
$$= m^1 \bmod n$$

(since we chose *ed* to be divisible by *(p-1)(q-1)* with remainder 1)

$= m$ (since $m<n$)

Authentication

- There is a clear need to “prove” the identity of a sender
- Insufficient options:
  - ID by IP # ?
  - Send secret password along with message ?
  - Choose a random number, R …



Bob computes

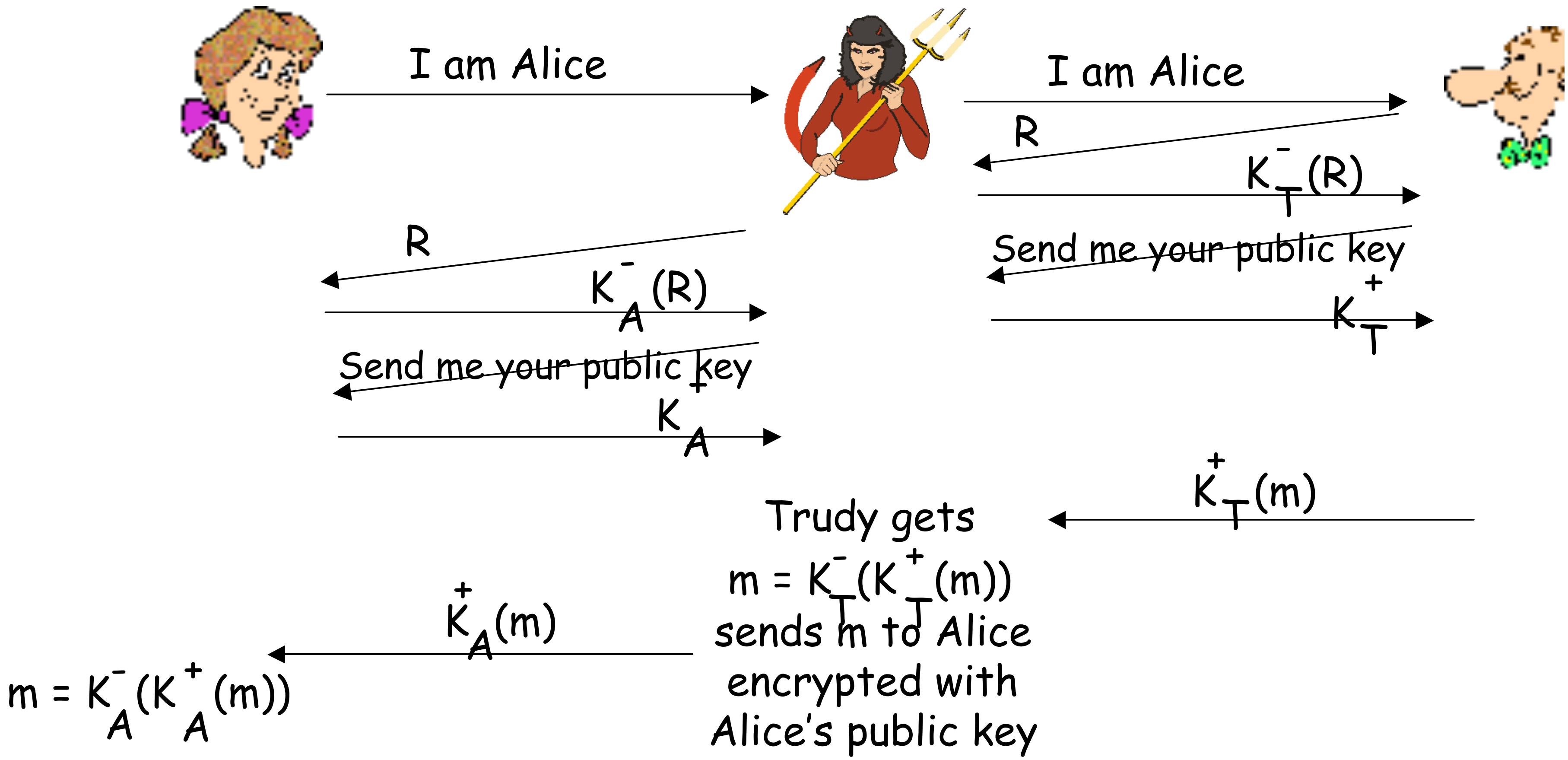
$$K_A^+(K_A^-(R)) = R$$

and knows only Alice could have the private key, that encrypted R such that

$$K_A^+(K_A^-(R)) = R$$

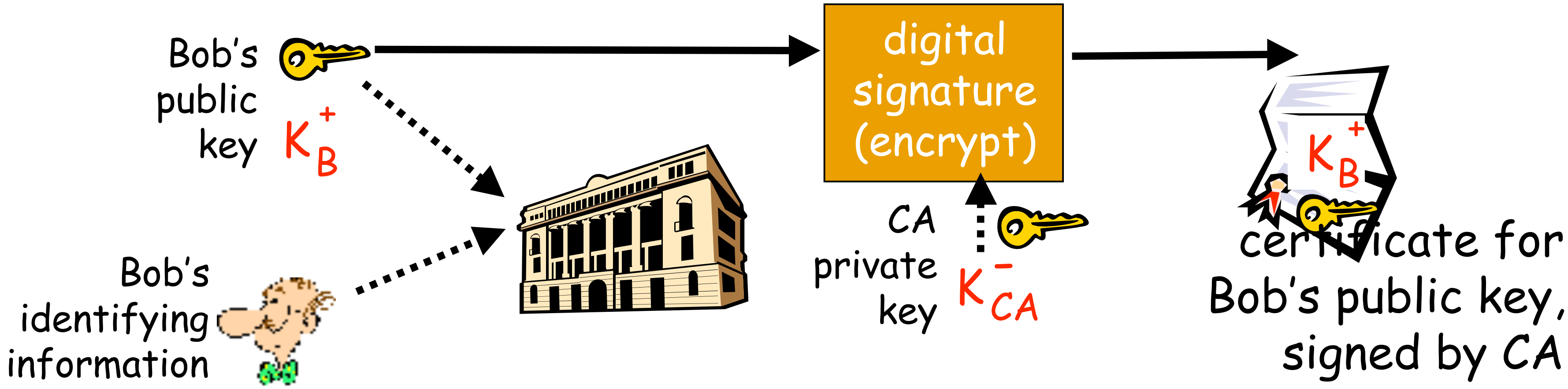
Man-in-the-middle Attack

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



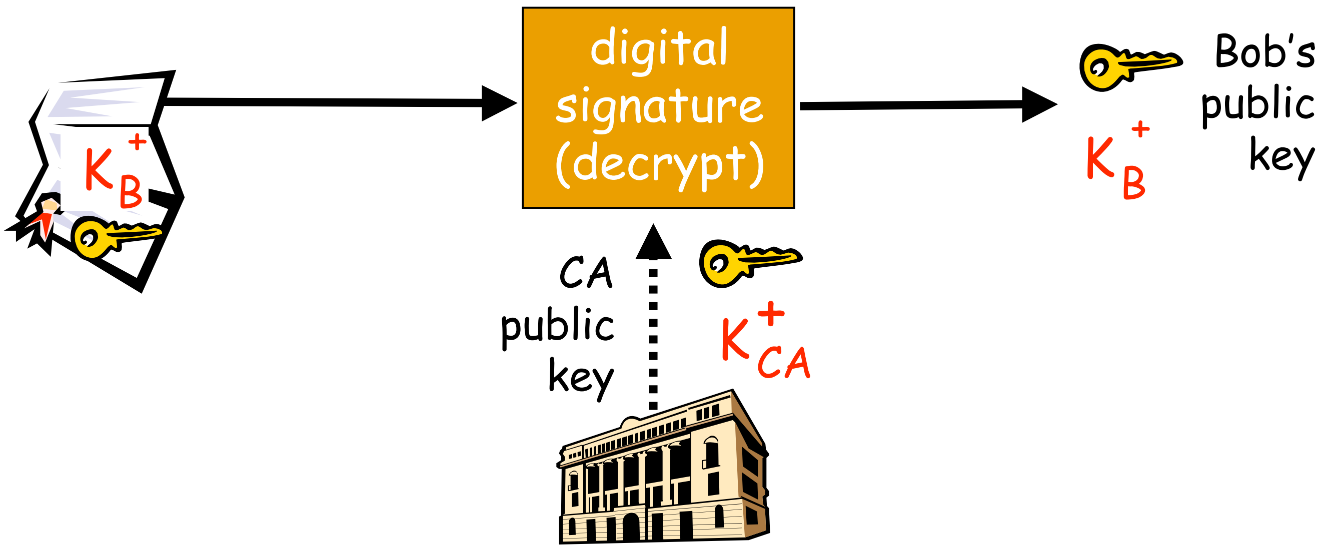
Certification Authorities

- Question: How do you “prove” that a key is really your key ?
- Solutions: Certification authority (CA) - binds public key to particular entity (for example: Bob).
- Bob registers its public key with CA.
  - Bob provides “proof of identity” to CA.
  - CA creates certificate binding Bob to its public key.
  - Certificate containing Bob’s public key digitally signed by CA – CA says “this is Bob’s public key”



Certification Authorities (cont.)

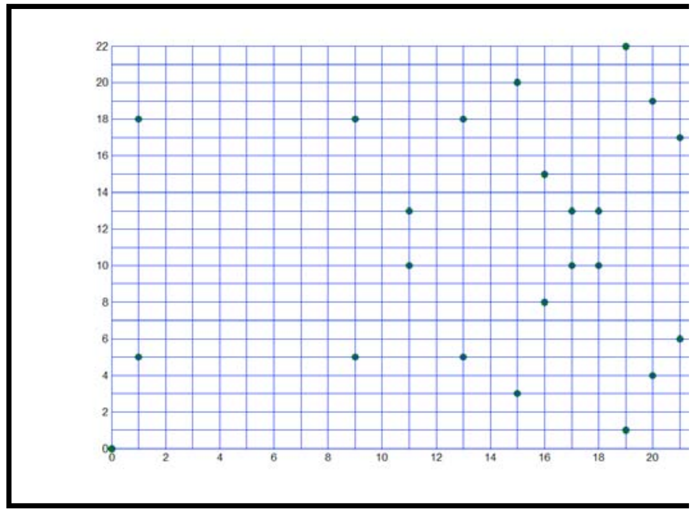
- When Alice wants Bob's public key:
  - gets Bob's certificate (Bob or elsewhere).
  - apply CA's public key to Bob's certificate, get Bob's public key



What is an Elliptic Curve?

In GF(p) an ordinary elliptic curve E suitable for elliptic curve cryptography is defined by the set of points (x; y) that satisfy the equation :

$$y^2 = (x^3 + ax + b) \bmod p$$

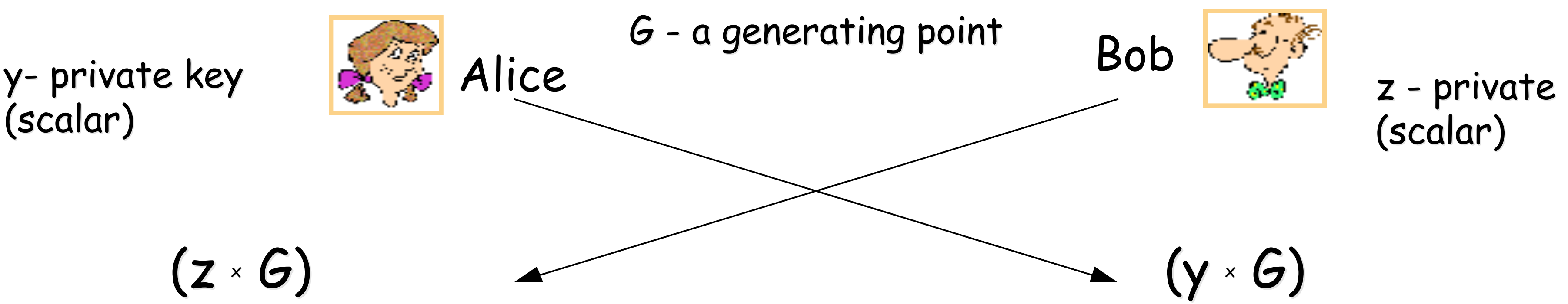


Using ECC (Elliptic Curve Cryptography) the Discrete-log problem takes the following form:

P, Q: Points on the curve		For a given P and Q, where P = Y × Q, there is no available algorithm to recover y
Y:	Large scalar (e.g 160)	

Diffie-Hellman Public Key Distribution Using ECC

- Why use Elliptic Curve Cryptography (ECC)?
- Calculations take less time, less memory and less hardware
- We use 160 bits (instead of 1024 bits used not in EC modula exponentiation, e.g. DH over a prime) and still retain the sam "security strength"



$$y \times (z \times G) = (y \times z) \times G = Z \times (y \times G)$$

Point-by-scalar multiplication is the core!

Prior Work: Key Pre-distribution Schemes

- Loading keys into sensor nodes prior to deployment
- Two nodes find a common key between them after deployment (a.k.a. "key discovery" phase)
- Possible solutions:
  - Master key – one key to all networks
    - (+) Minimal communications (low power consumption)
    - (+) Memory efficient, Key discovery is not really needed
    - (-) However, once key is compromised entire network is compromised
  - N-1 keys to each node
    - (+) Key discovery is not really needed
    - (-) Cannot add new nodes!
    - (-) Memory requirements are not practical (non-scalable)

Prior Work: Key Pre-distribution Schemes (cont.)

AICIP RESEAR

- Random Key Predistribution

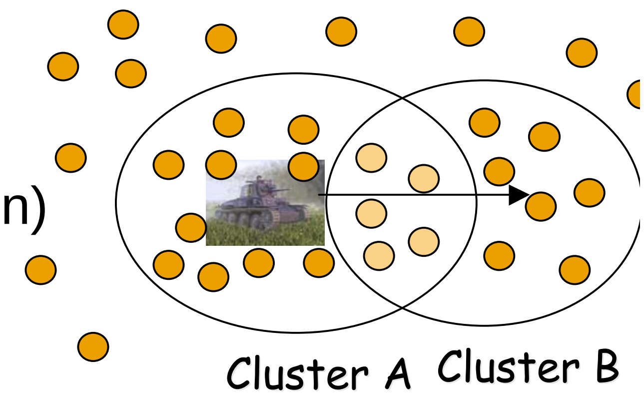
  Each node is provided with a subset of a large key pool

  - (+) Ability to add nodes after deployment
  - (+) Lower network compromise with captured nodes
  - (-) Key discovery is needed

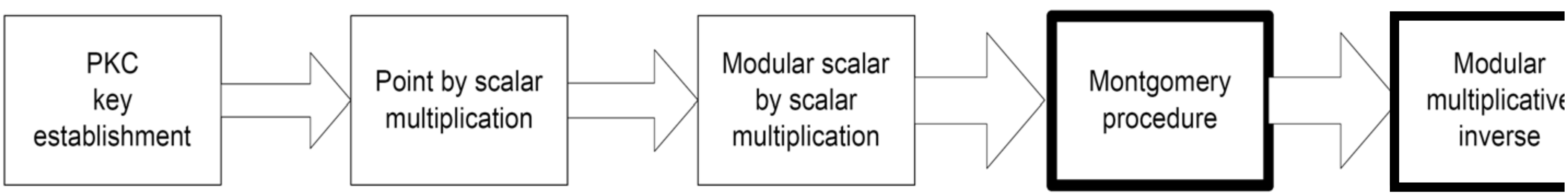
- Fundamental limitations to random key pre-distribution schemes:
  - Scalability – the memory, network size
  - Communication framework - finding nodes sharing keys
  - Cryptographic robustness – inherently offer “statistical” security, whic always questionable

Potential Solutions

- How to reduce the amount of point-by-scalar multiplication?
  - Self-certified key generation
  - Fixed key generation (1 multiplication)
  - Ephemeral key generation (2 multiplication)
    - Off-loading 1 multiplication to neighbors
  - Group key generation



- How to mitigate denial-of-service attack?
- How to reduce the complexity of point-by-scalar multiplication?



Reference

- O. Arazi, H. Qi, “A Public-Key Cryptographic Methodology for Denial of Service Mitigation in Wireless Sensor Networks," in Proc. IEEE SECON 2007.
- O. Arazi, H. Qi, “An Efficient Computation of Inverse Multiplicative Operations," IEEE Trans. on Computers, 2008.
- O.Arazi, H. Qi, "Load-Balanced Key Establishment Methodologies in Wireless Sensor Networks," International Journal of Sensor Networks (IJSN) (Special issu on Security for Sensor Networks), Vol. 1, No. 2, April 2006.
- O. Arazi, D. Rose, H. Qi, I. Elhanany B. Arazi, "Self-Certified Public Key Generation on the Intel Mote 2 Sensor Network Platform" 3rd Annual IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), Sept., 2006, Reston, VA.
- O. Arazi and H. Qi, "Self-Certified Group Key Generation for Ad Hoc Clusters in Wireless Sensor Networks," in Proc. of the 14th IEEE International Conference o Computer Communications and Networks (ICCCN), San Diego, CA, Oct. 17-19, 2005.
- B.Arazi, I. Elhanany, O. Arazi, and H. Qi, "Revisiting Public Key Cryptography for Wireless Sensor Network," IEEE Computer Magazine, pp. 85-87, Nov. 2005.