CSE 484 (Winter 2010)

Asymmetric Cryptography

Tadayoshi Kohno

Thanks to Dan Boneh, Dieter Gollmann, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

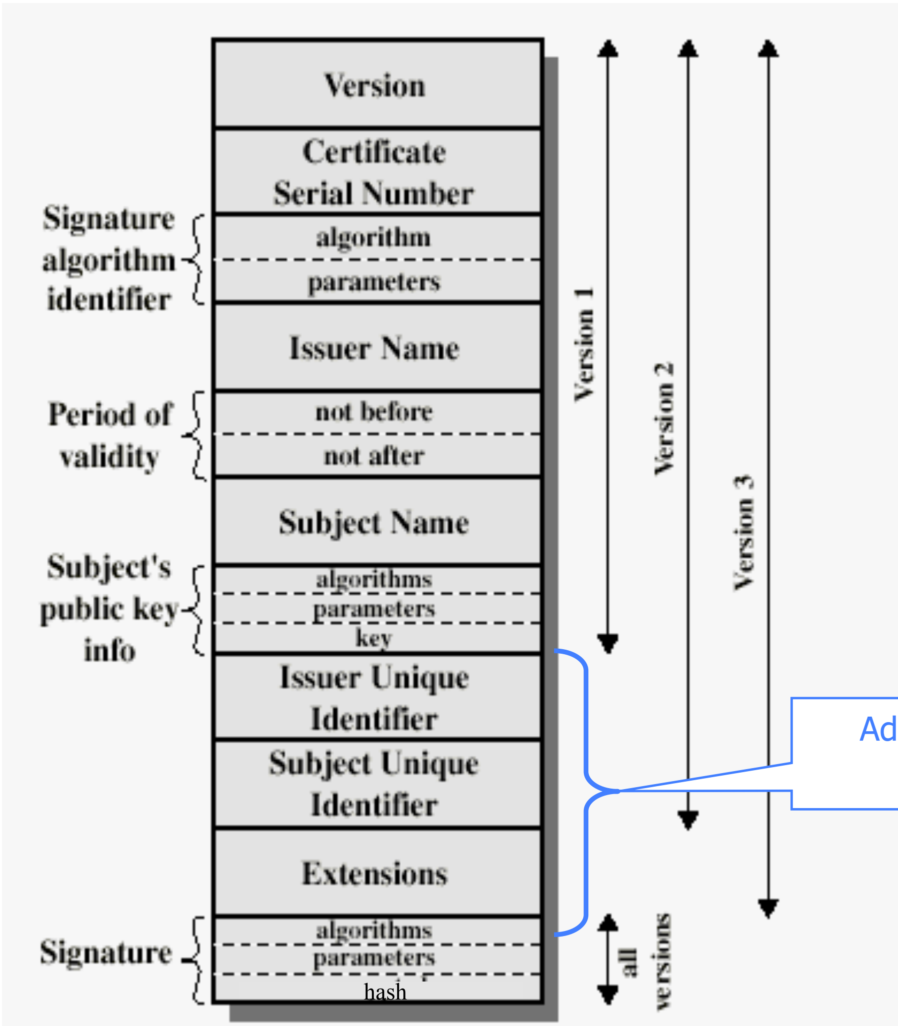
Goals for Today

- PKI
- Key Establishment

X.509 Authentication Service

- Internet standard (1988 onward)
- Specifies certificate format
  - X.509 certificates are used in IPSec and SSL/TLS
- Specifies certificate directory service
  - For retrieving other users' CA-certified public keys
- Specifies a set of authentication protocols
  - For proving identity using public-key signatures
- Does <u>not</u> specify crypto algorithms
  - Can use it with any digital signature scheme and hash function, but hashing is required before signing

X.509 Certificate



Added in X.509 versions 2 and 3 to address usability and security problems

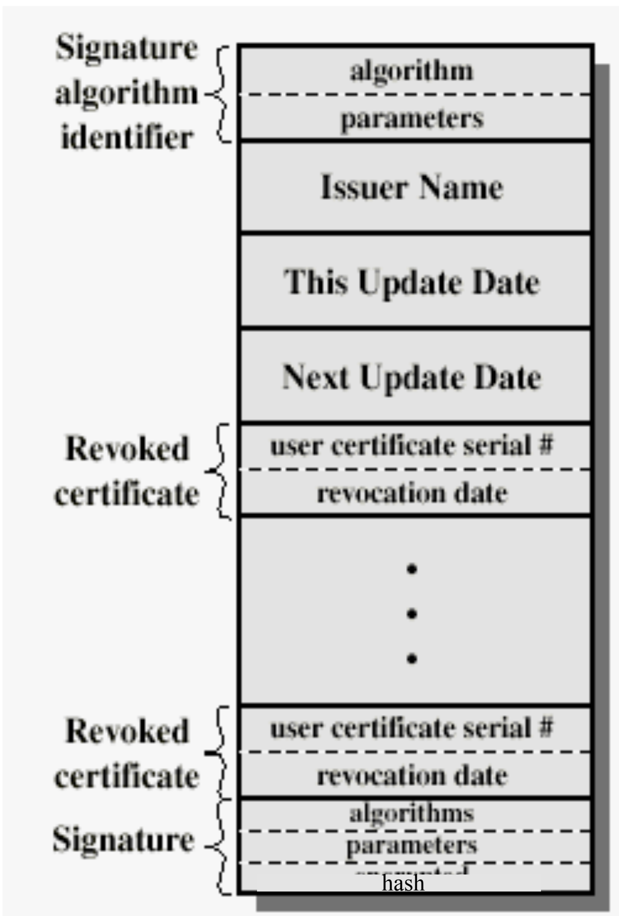
Certificate Revocation

- Revocation is <u>very</u> important
- Many valid reasons to revoke a certificate
  - Private key corresponding to the certified public key has been compromised
  - User stopped paying his certification fee to this CA and CA no longer wishes to certify him
  - CA's private key has been compromised!
- Expiration is a form of revocation, too
  - Many deployed systems don't bother with revocation
  - Re-issuance of certificates is a big revenue source for certificate authorities

Certificate Revocation Mechanisms

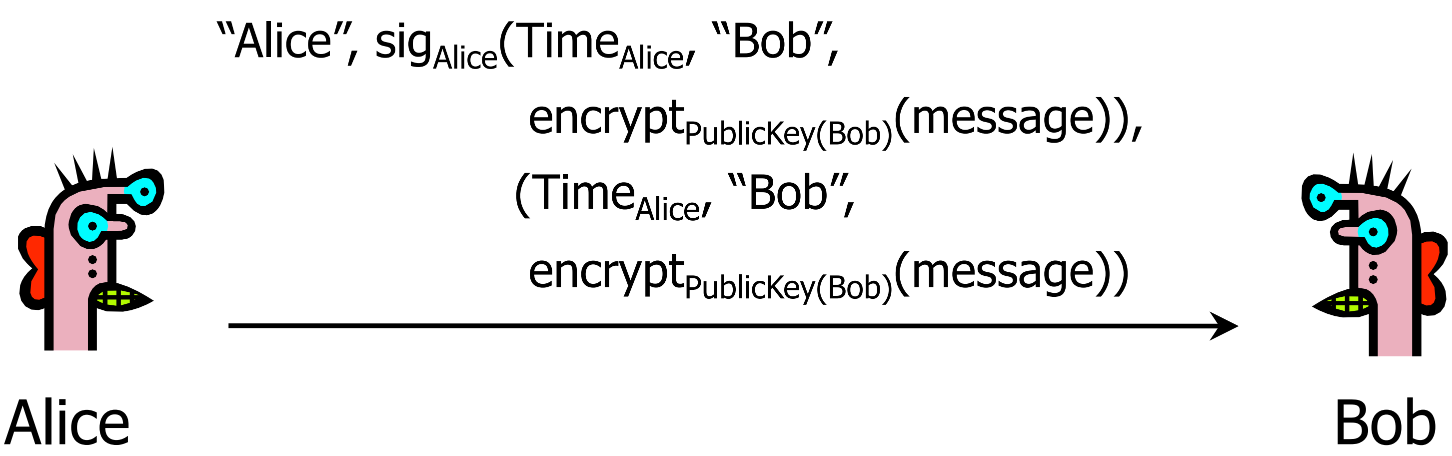
- Online revocation service
  - When a certificate is presented, recipient goes to a special online service to verify whether it is still valid
    - Like a merchant dialing up the credit card processor
- Certificate revocation list (CRL)
  - CA periodically issues a signed list of revoked certificates
    - Credit card companies used to issue thick books of canceled credit card numbers
  - Can issue a "delta CRL" containing only updates

X.509 Certificate Revocation List



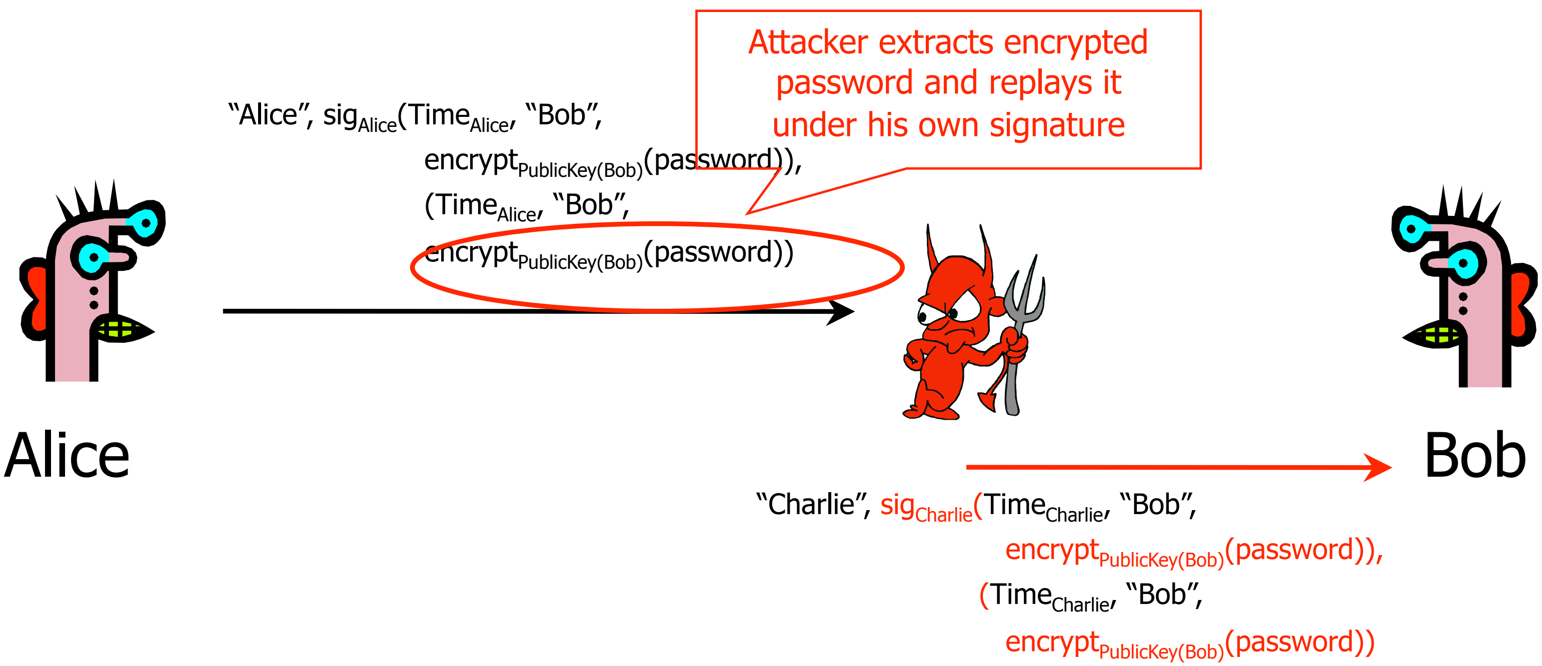
Because certificate serial numbers must be unique within each CA, this is enough to identify the certificate

X.509 Version 1



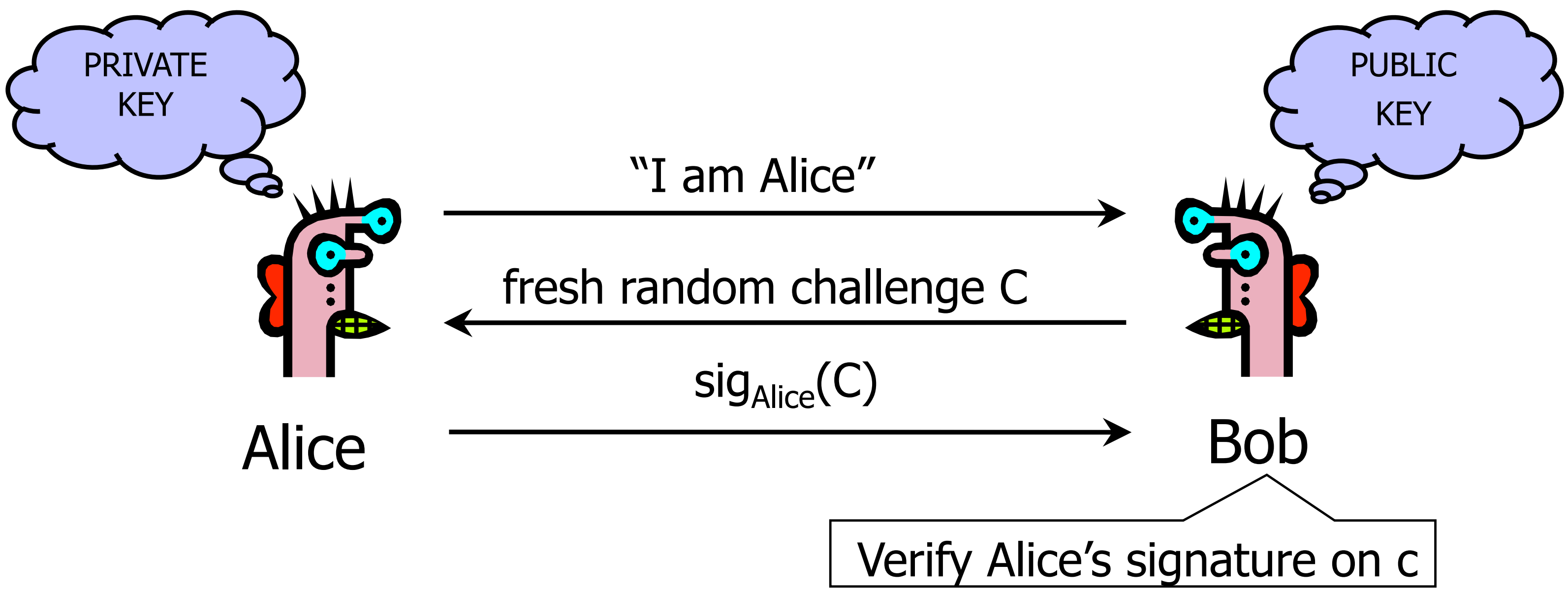
- Encrypt, then sign
  - Goal: achieve both confidentiality and authentication
  - E.g., encrypted, signed password for access control (for next slide: assume one password for whole system)
- Does this work?

Attack on X.509 Version 1



- Receiving encrypted password under signature does <u>not</u> mean that the sender actually knows the password!

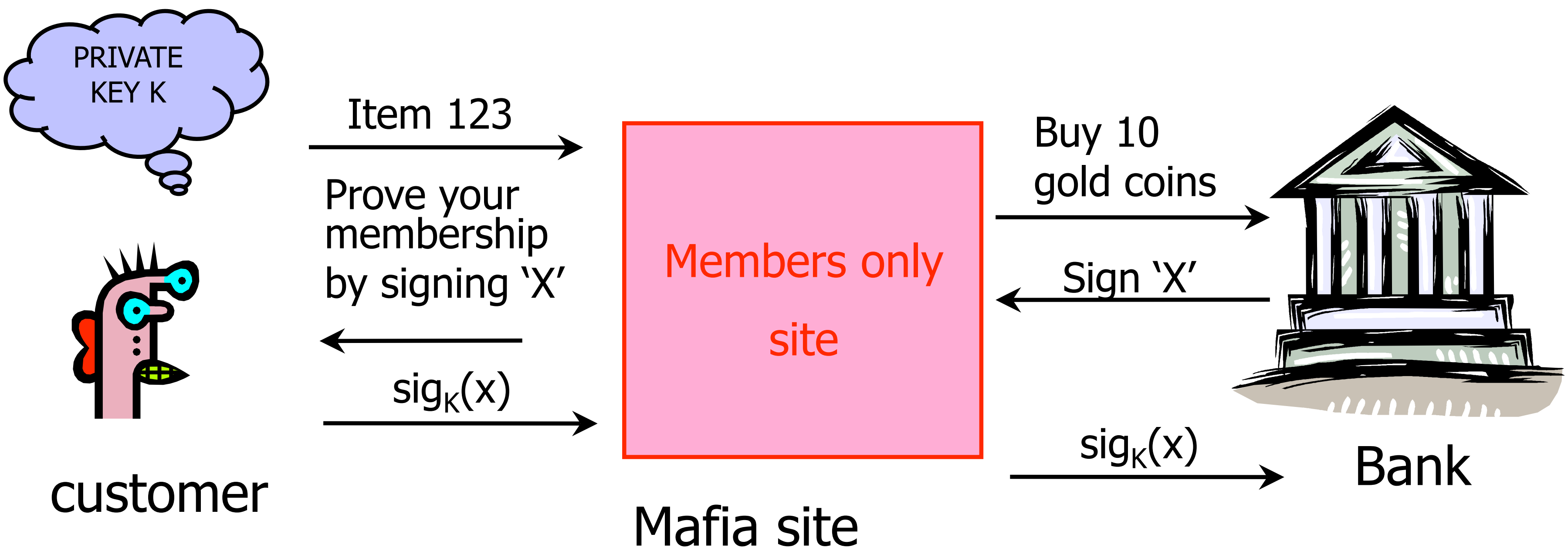
Authentication with Public Keys



1. Only Alice can create a valid signature
2. Signature is on a fresh, unpredictable challenge

Potential problem: Alice will sign anything

Mafia-in-the-Middle Attack [from Anderson's book]



One key recommendation: Don't use same public key / secret key pair for multiple applications. (Or make sure messages have different formats across applications.)

Secure Sessions

- Secure sessions are among the most important applications in network security
  - Enable us to talk securely on an insecure network
- Goal: secure bi-directional communication channel between two parties
  - The channel must provide confidentiality
    - Third party cannot read messages on the channel
  - The channel must provide authentication
    - Each party must be sure who the other party is
  - Other desirable properties: integrity, protection against denial of service, anonymity against eavesdroppers

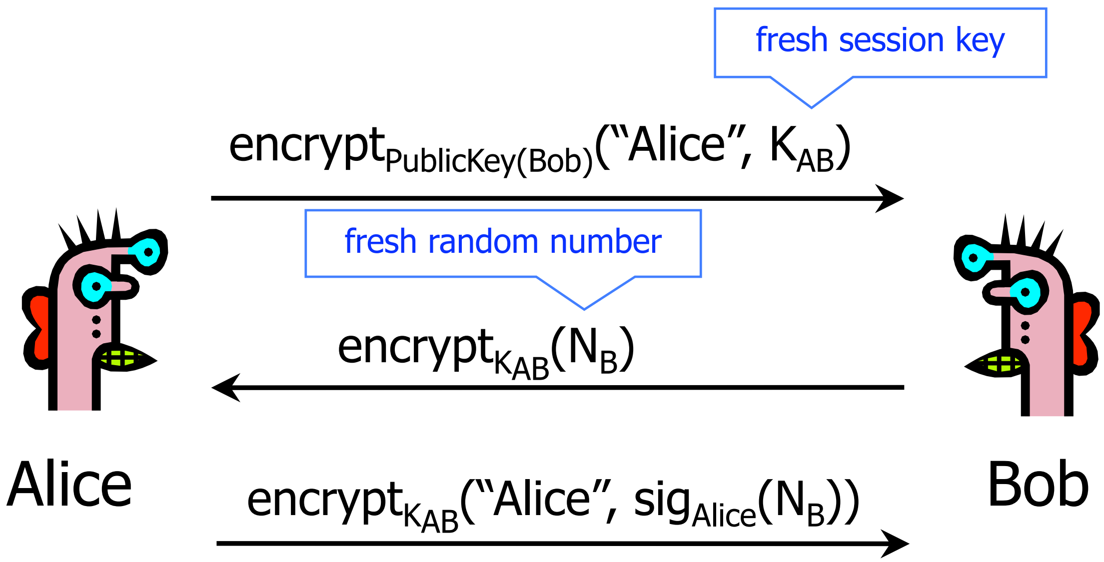
Key Establishment Protocols

- Common implementation of secure sessions:
  - Establish a secret key known only to two parties
  - Then use block ciphers for confidentiality, HMAC for authentication, and so on
- Challenge: how to establish a secret key
  - Using only <u>public</u> information?
  - Even if the two parties share a long-term secret, a fresh key should be created for each session
    - Long-term secrets are valuable; want to use them as sparingly as possible to limit exposure and the damage if the key is compromised
    - (Background: For N parties, there are N choose 2 = N*(N-1)/2 pairs of parties.)

Key Establishment Techniques

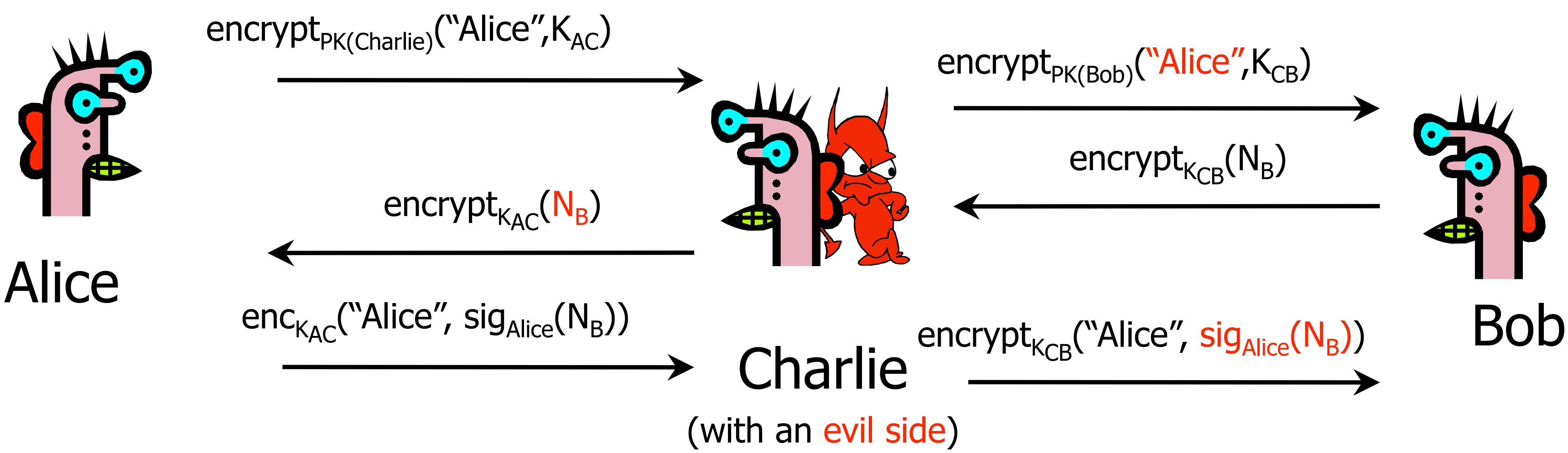
- Use a trusted key distribution center (KDC)
  - Every party shares a pairwise secret key with KDC
  - KDC creates a new random session key and then distributes it, encrypted under the pairwise keys
    - Example: Kerberos
- Use public-key cryptography
  - Diffie-Hellman authenticated with signatures
    - Example: IKE (Internet Key Exchange)
  - One party creates a random key, sends it encrypted under the other party's public key
    - Example: TLS (Transport Layer Security)

Early Version of SSL (Simplified)



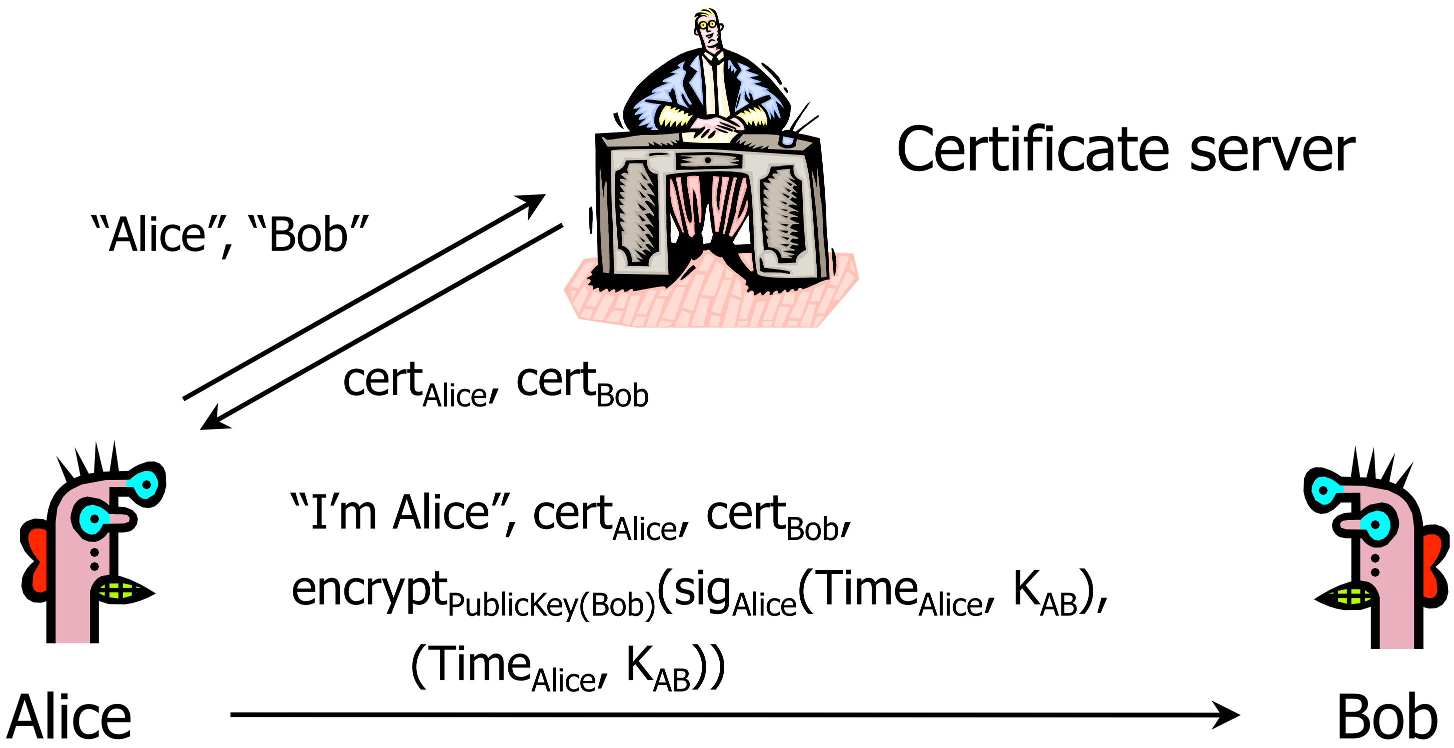
- Bob's reasoning: I must be talking to Alice because…
  - Whoever signed N_B knows Alice's private key… Only Alice knows her private key… Alice must have signed N_B… N_B is fresh and random and I sent it encrypted under K_{AB}… Alice could have learned N_B only if she knows K_{AB}… She must be the person who sent me K_{AB} in the first message...

Breaking Early SSL



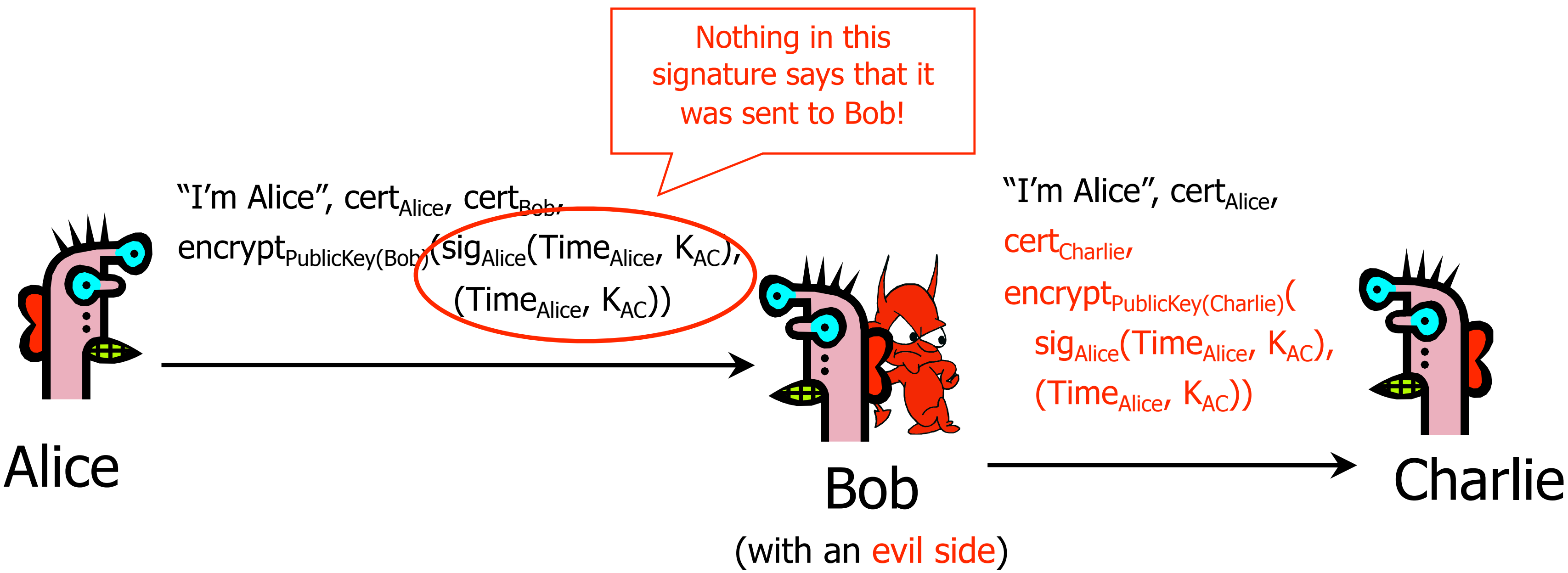
- Charlie uses his legitimate conversation with Alice to impersonate Alice to Bob
  - Information signed by Alice is not sufficiently explicit

Denning-Sacco Protocol



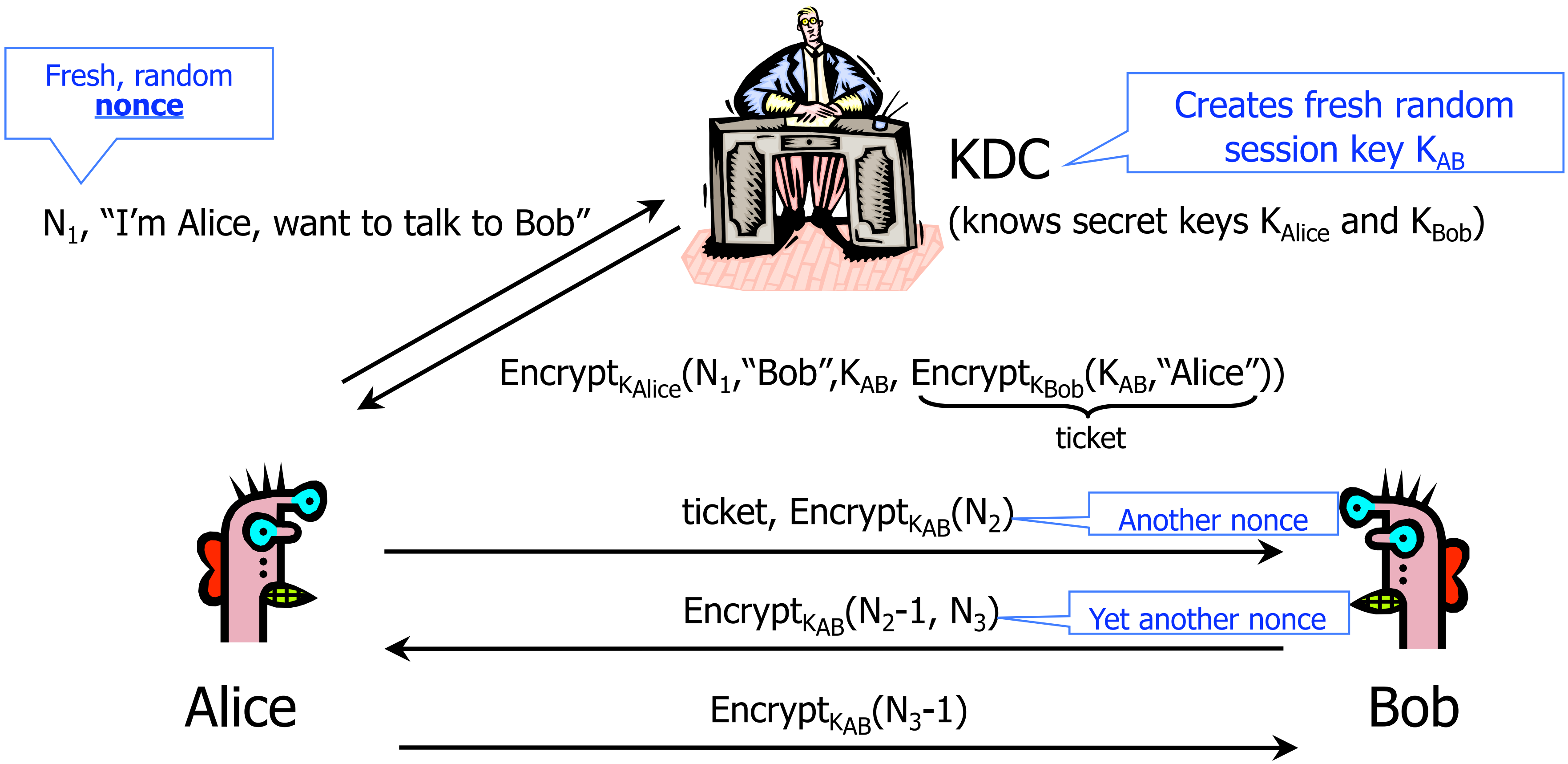
- Goal: establish a new shared key K_{AB} with the help of a trusted certificate service

Attack on Denning-Sacco



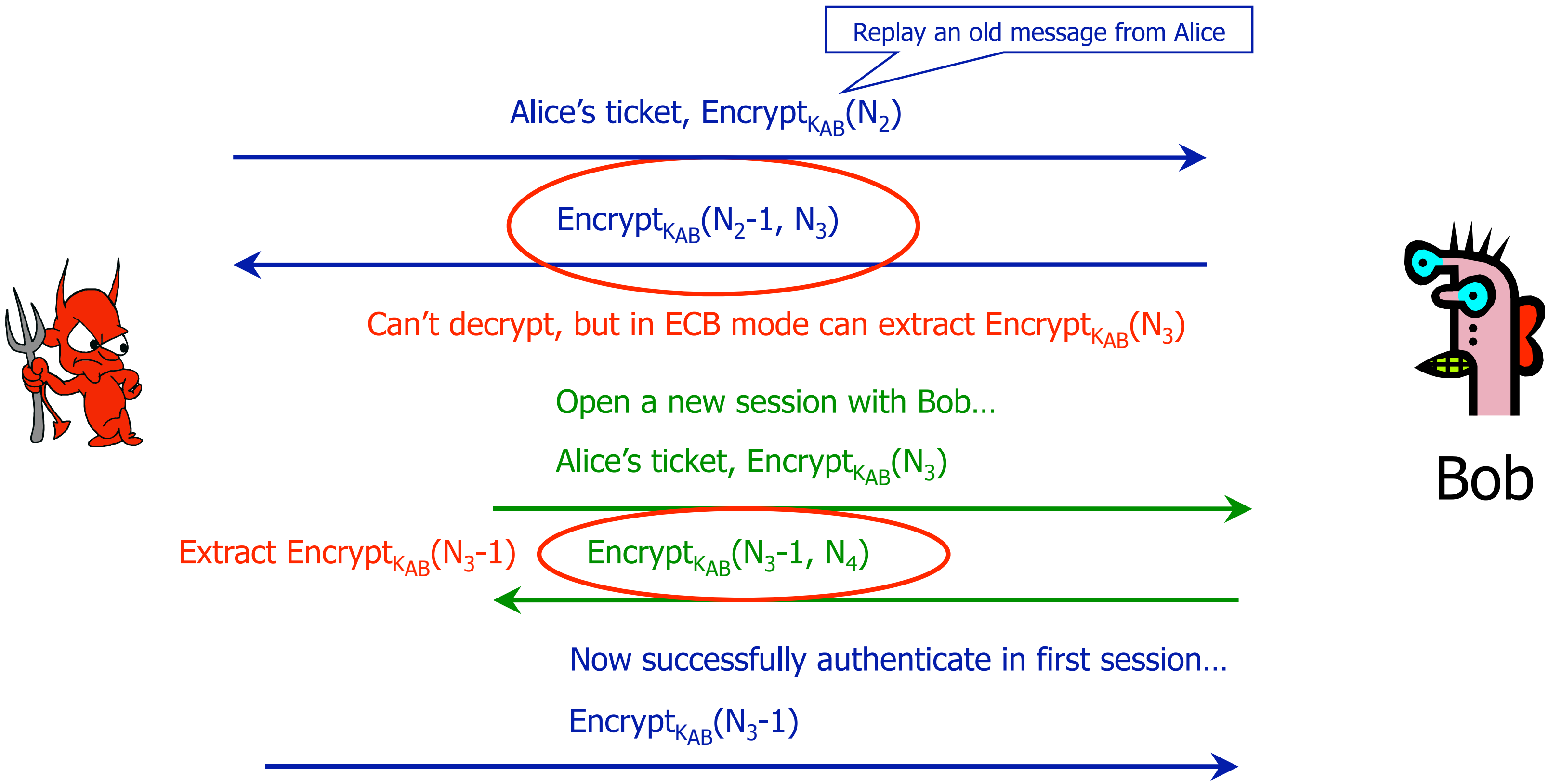
- Alice's signature is insufficiently explicit
  - Does not say to whom and why it was sent
- Alice's signature can be used to impersonate her

Private-Key Needham-Schroeder

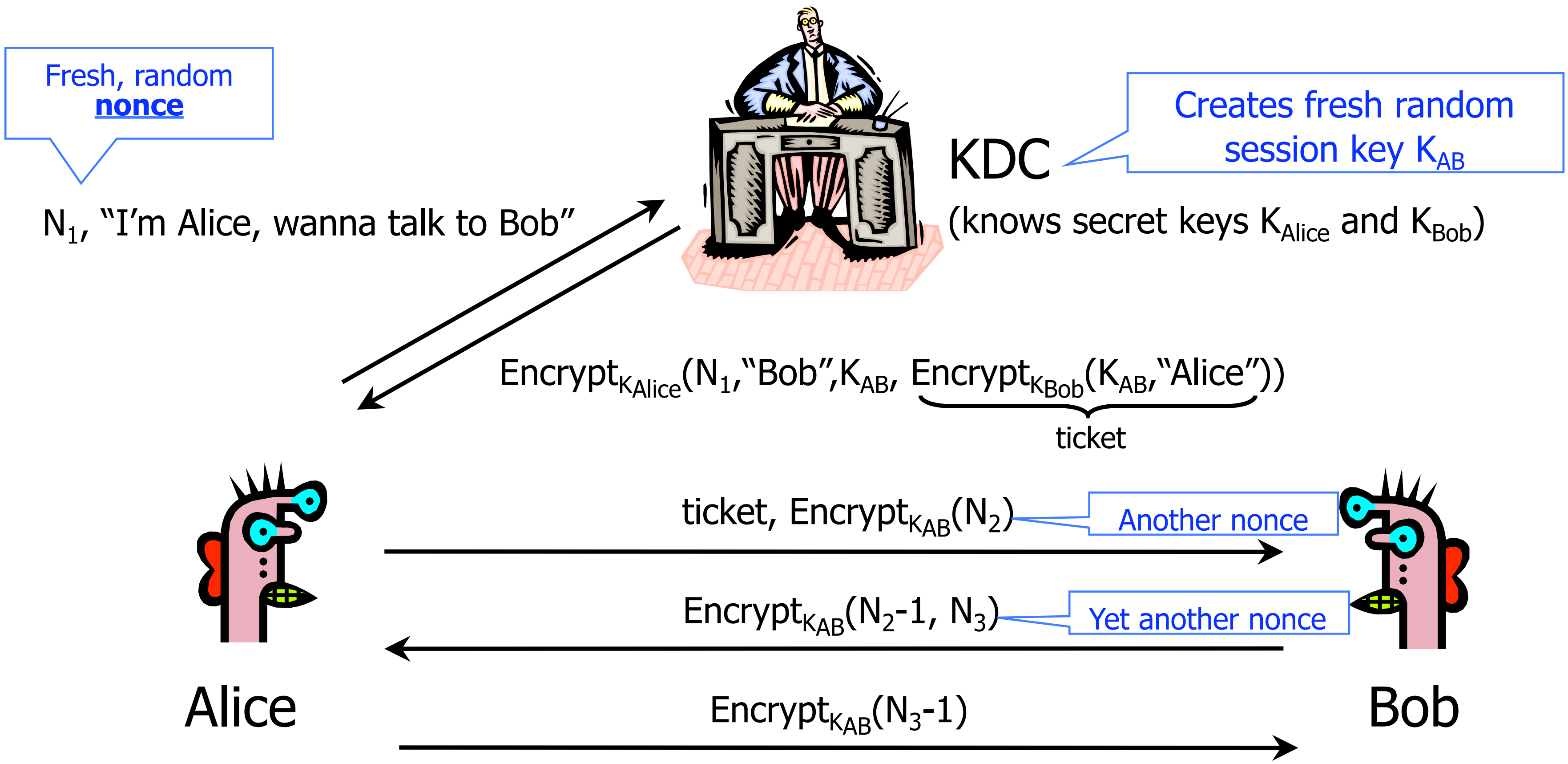


Reflection Attack

- Suppose symmetric encryption is in ECB/CBC mode…
  - (Easier to see with ECB mode, so assume that)



Private-Key Needham-Schroeder



- Another issue: If learn K_{AB} after session completes, then can re-use. (Solution: timestamps, nonces.)

Public-Key Needham-Schroeder

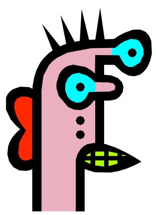
Alice's nonce

$\text{Encrypt}_{\text{PublicKey(Bob)}}(\text{“Alice”}, N_A)$ →

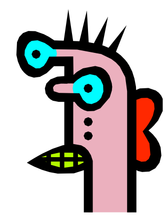
Bob's nonce

← $\text{Encrypt}_{\text{PublicKey(Alice)}}(N_A, N_B)$

$\text{Encrypt}_{\text{PublicKey(Bob)}}(N_B)$ →



Alice



Bob

Create new key from N_A and N_B, e.g., $N_A \oplus N_B$

Alice's reasoning:

- The only person who could know N_A is the person who decrypted 1st message
- Only Bob can decrypt message encrypted with Bob's public key
- Therefore, Bob is on the other end of the line

Bob is authenticated!

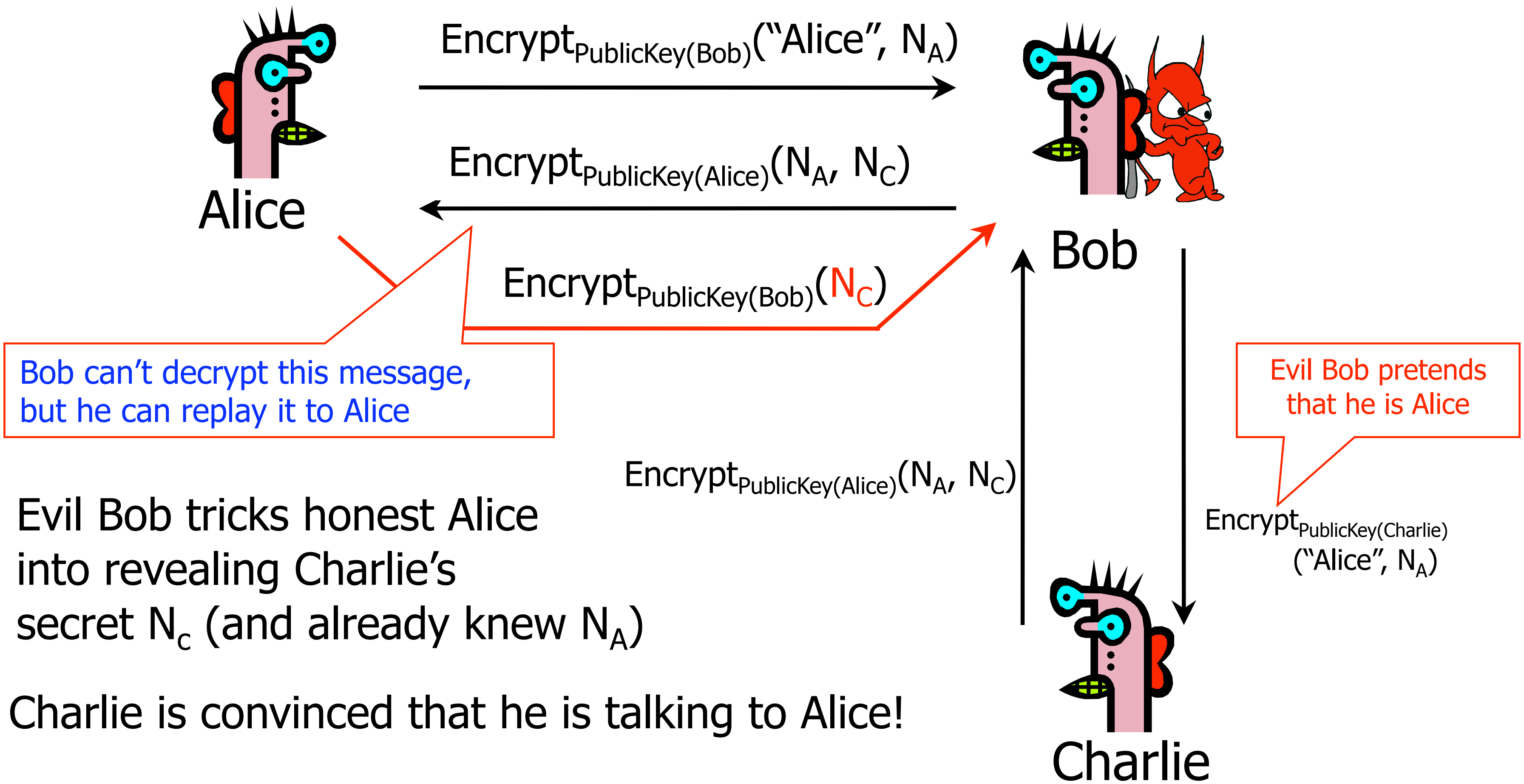
Bob's reasoning:

- The only way to learn N_B is to decrypt 2nd message
- Only Alice can decrypt 2nd message
- Therefore, Alice is on the other end

Alice is authenticated!

Attack on Needham-Schroeder

[published by Gavin Lowe]



Evil Bob tricks honest Alice into revealing Charlie's secret N_c (and already knew N_A)

Charlie is convinced that he is talking to Alice!

Lessons of Needham-Schroeder

- This is yet another example of design challenges
  - Alice is correct that Bob must have decrypted $\text{Encrypt}_{\text{PublicKey(Bob)}}(\text{“Alice”}, N_A)$, but this does not mean that $\text{Encrypt}_{\text{PublicKey(Alice)}}(N_A, N_B)$ came from Bob
- It is important to realize limitations of protocols
  - The attack requires that Alice willingly talk to attacker
    - Attacker uses a legitimate conversation with Alice to impersonate Alice to Charlie