Fundamentals of Information & Network Security
ECE 471/571

Lecture #30,31: Cryptographic Authentication Protocols

Instructor: Ming Li
Dept of Electrical and Computer Engineering
University of Arizona

Security Protocol

- An agreement between communication parties about the process and the format of security bootstrap, authentication, key establishment, encryption/hashing algorithm and parameter negotiation, etc.

- Typically include
  - Authentication handshake
  - Session key negotiation, algorithm/parameter negotiation
  - Data encryption and/or integrity protection

- You do not need to design your own crypto algorithms, but you may need to design a security protocol.

Cryptographic Authentication

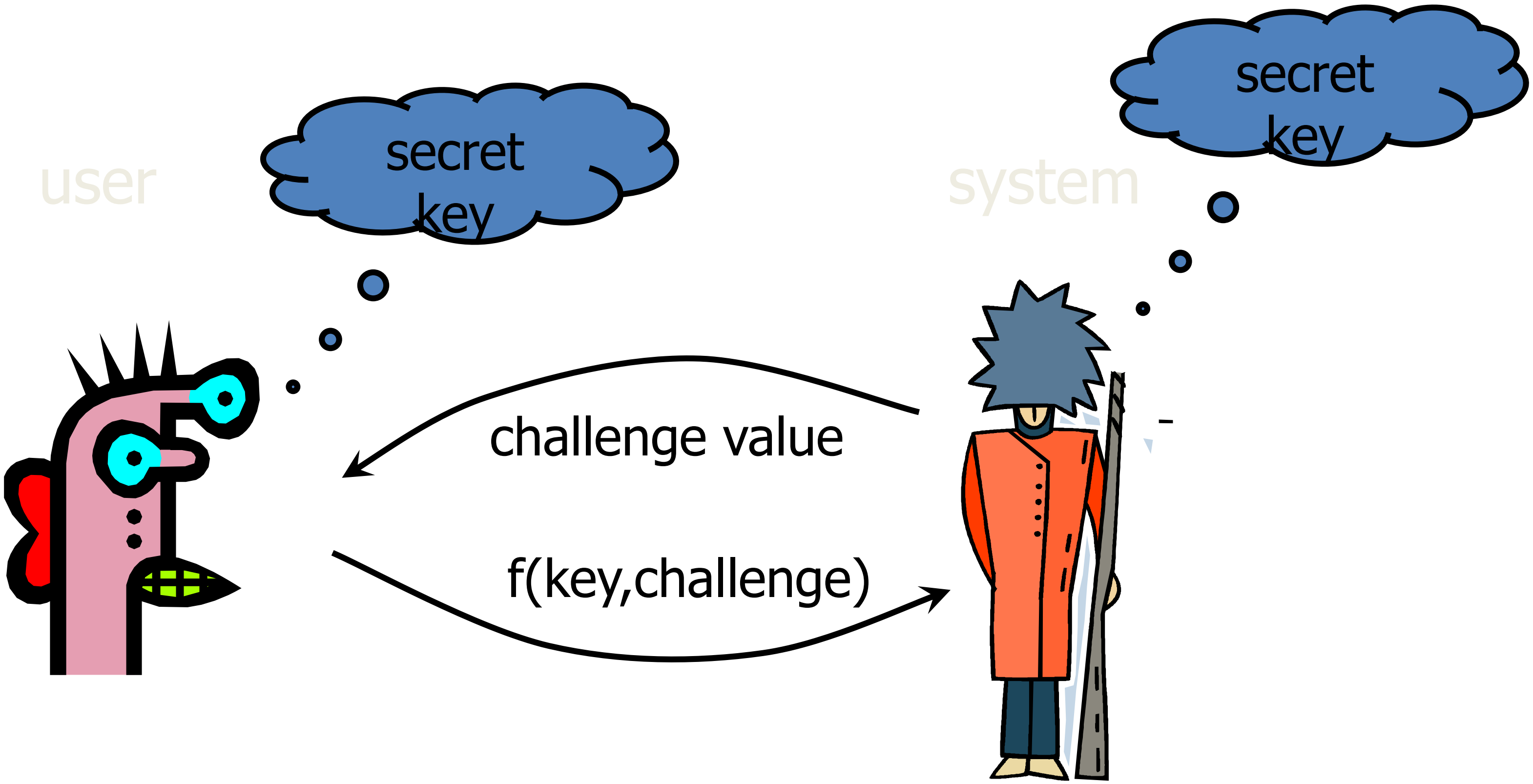
- Bob sends Alice a random number.
- Alice uses its cryptographic key to perform a cryptographic operation on the number.
- Alice sends the cryptographic result to Bob.
- Bob verifies the result.

Challenge-response process

Challenge/response: A bad idea

- Alice sends name and password in clear (across network) to Bob
- Bob verifies name and password and communication proceeds

Challenge-Response



Why is this better than a password over a network?

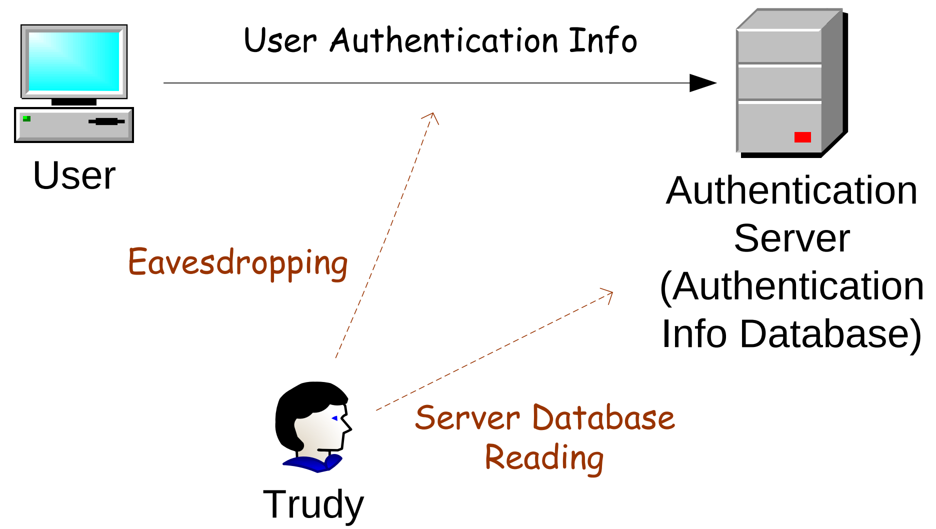
Challenge-Response Authentication

- User and system share a secret key
- Challenge: system presents user with some string
- Response: user computes response based on secret key and challenge
  - Secrecy: difficult to recover key from response
    - One-way hashing or symmetric encryption work well
  - Freshness: if challenge is fresh and unpredictable, attacker on the network cannot replay an old response
    - For example, use a fresh random number for each challenge
- Good for systems with pre-installed secret keys

Passwords as Cryptographic Keys

- Secret Key
  - Hash a password to get a DES key
- Public Key
  - Use a password as the seed for a random number generator to create a private/public key pair
  - Use a password to encrypt a private key

Eavesdropping and Server Database Reading



- Public key cryptography is secure against both eavesdropping and server database reading.
- Password or secret key cryptography is resilient to either one but not both (why?)

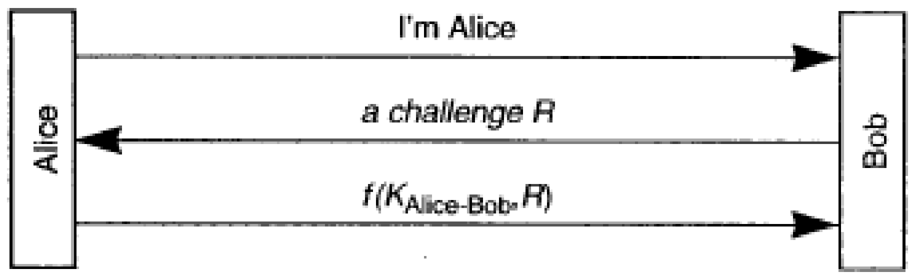
Security Bootstrap

- Shared secret
  - Password (for human users)
  - Pre-shared key (between firewalls)
  - Ticket by KDC (among a large number of participates)

- Public key
  - Manually configured
  - Encrypted by a password (for human users)
  - Certificate by the client’s private key
  - Certificate by CA

One-way Authentication by Shared Secret (Protocol 1)

- Let K be the shared secret.

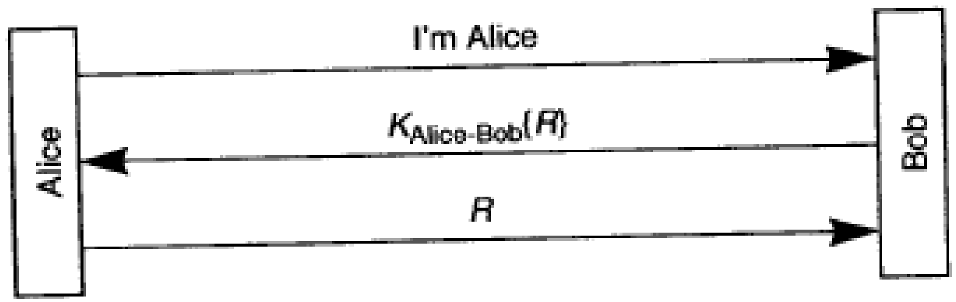


Protocol 11-1. Bob authenticates Alice based on a shared secret $K_{Alice\text{-}Bob}$

- f(K,R) is E_K\{R\} or hash(K,R)
- One way authentication only: Bob authenticates Alice.
  - off-line password-guessing attack
  - Database reading

Protocol 2

- Let K be the shared secret.



Protocol 11-2. Bob authenticates Alice based on a shared secret key $K_{Alice\text{-}Bob}$

- Issues: same as Protocol 1
- If R is a recognizable number, then it does mutual authentication.
- Make R recognizable but with limited lifetime. For instance, timestamp. It however requires clock synchronization.

Protocol 3

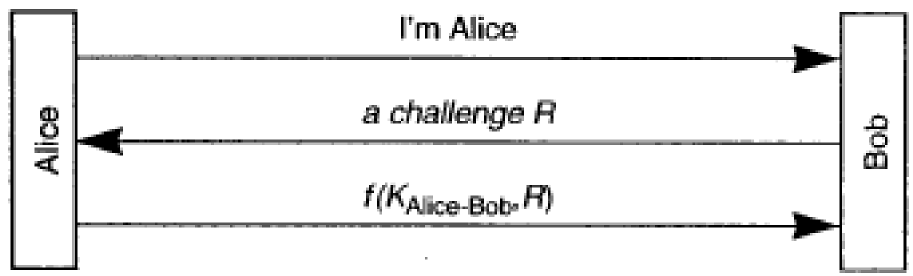
- Let K be the shared secret.

Protocol 11-3. Bob authenticates Alice based on synchronized clocks and a shared secret $K_{Alice-Bob}$

- Advantage:
  - It can be easily added into an existing protocol.
  - It saves two messages.
  - Bob is *stateless* *.
- Issues:
  - replay attack (single server, multiple server),
  - reset-clock attack

Protocol 4

- Let K be the shared secret, f(K,R) is hash(K,R)

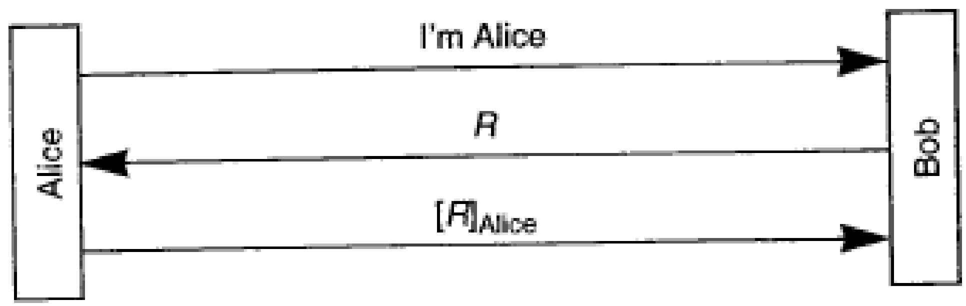


Protocol 11-1. Bob authenticates Alice based on a shared secret $K_{\text{Alice-Bob}}$

Protocol 11-4. Bob authenticates Alice based on hashing a high-resolution time and a shared secret $K_{\text{Alice-Bob}}$

One-Way Authentication by Public Key (Protocol 5 and 6)

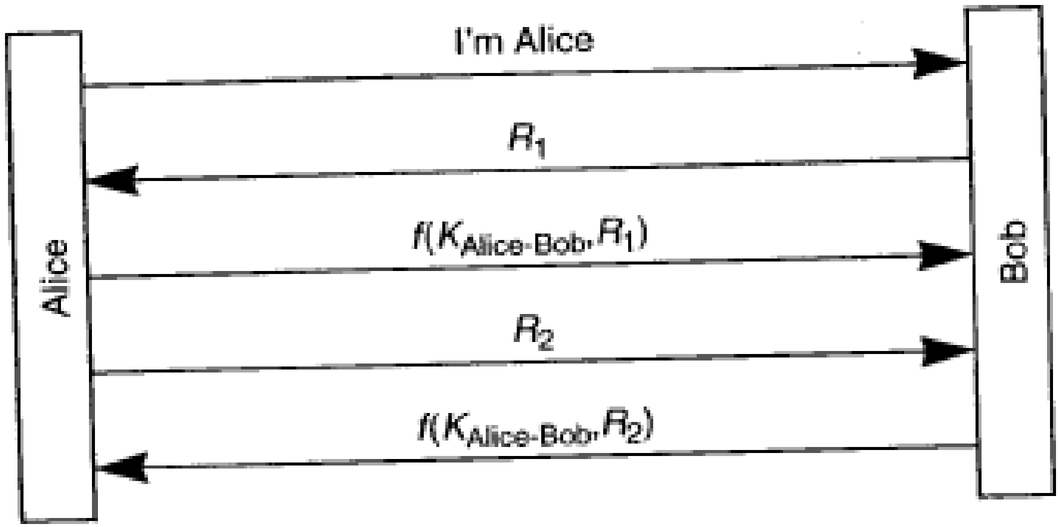
- Alice has her private key. Bob has Alice's public key.



Protocol 11-5. Bob authenticates Alice based on her public key signature

- Protocol 6:
    - Bob->Alice: $\{R\}_{Alice}$
    - Alice->Bob: R
- Attacks exploiting same-key different-uses
- Offline password-guessing attack.

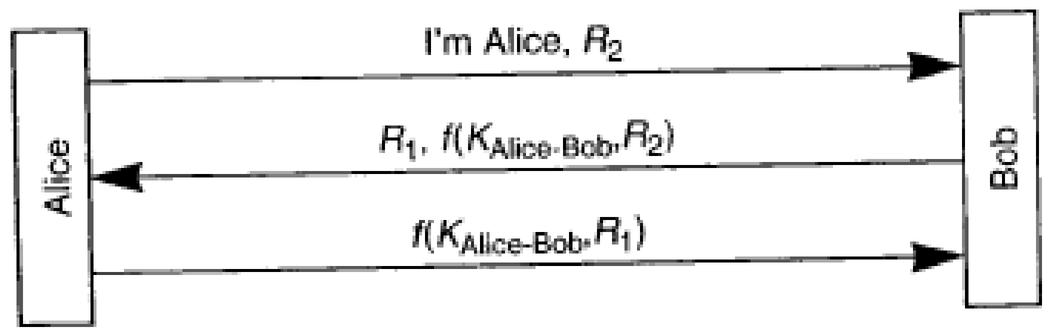
Mutual Authentication by Secret Key (protocol 7)



Protocol 11-7. Mutual authentication based on a shared secret $K_{Alice\text{-}Bob}$

- Issues: inefficient

Protocol 8

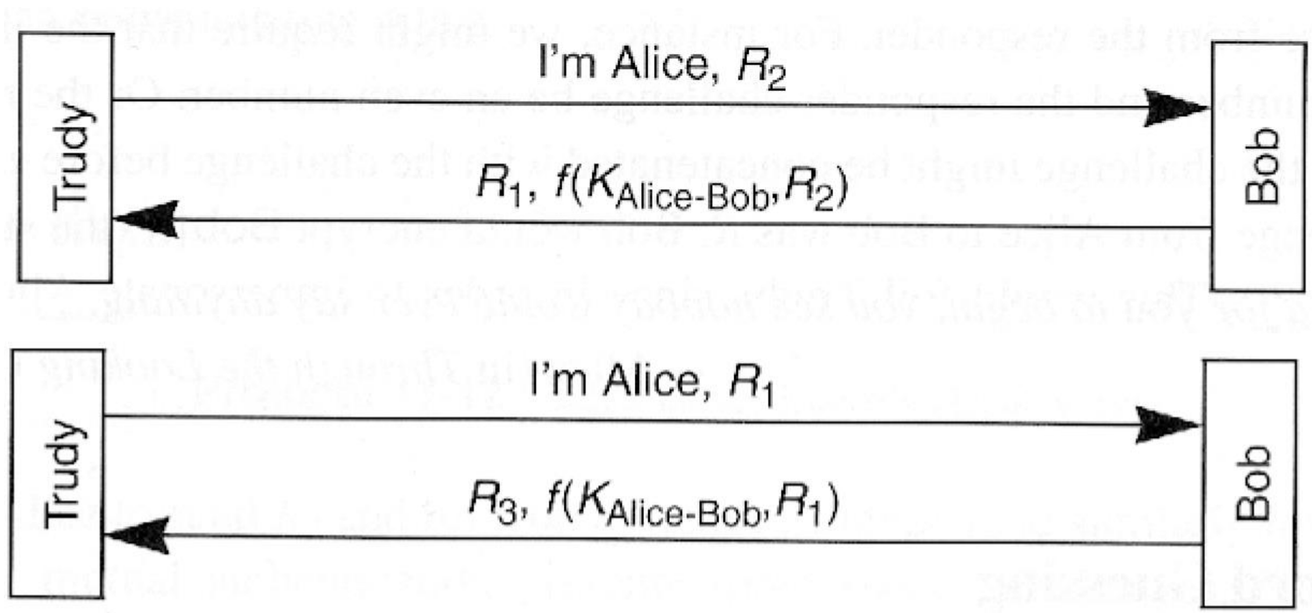


Protocol 11-8. Optimized mutual authentication based on a shared secret $K_{Alice\text{-}Bob}$

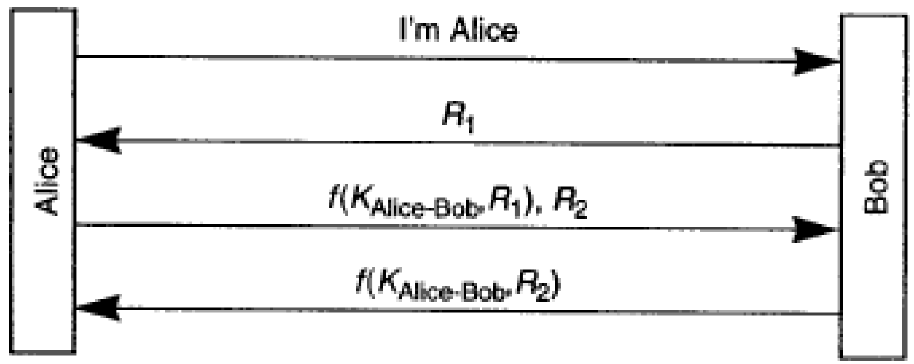
- Issues: reflection attack, password guessing
- Rule: Alice and Bob should not do exactly the same thing (use different keys, different challenges).
  - For example, Bob's key might be – Key_AB+1, or K_AB xor F0F0F0F0…
  - Different challenges: Initiator challenge be odd number, responder's challenge be even number, Alice's response uses R1+1, etc.

Reflection Attack

- Trudy opens 1st session to Bob
- Trudy opens 2nd session to Bob/Alice in order to get information needed to complete 1st session.



Protocol 11

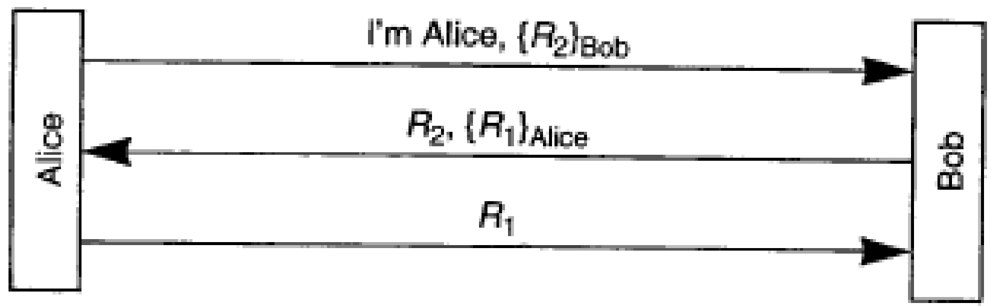


Protocol 11-11. Less optimized mutual authentication based on a shared secret $K_{\text{Alice-Bob}}$

- Less vulnerable to password guessing. The attack has to eavesdrop.
- Rule: The initiator should be the first to prove its identity (the assumption is that the initiator is more likely the bad guy).

Mutual Authentication by Public Key (Protocol 12)

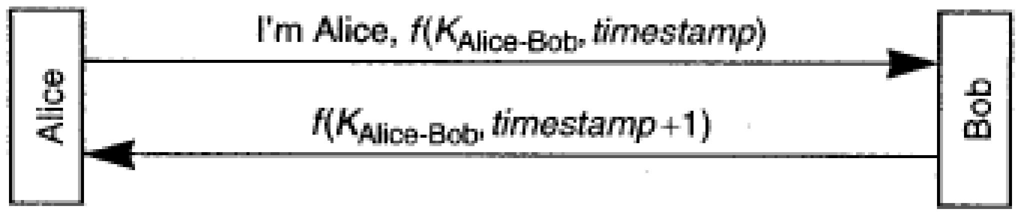
- Alice (Bob) know her (his) own private key and the other party's public key.



Protocol 11-12. Mutual authentication with public keys

- Variant
  - Alice → Bob: I'm Alice, R2
  - Bob → Alice: $[R2]_{Bob}$, R1
  - Alice → Bob: $[R1]_{Alice}$

Mutual Authentication by Timestamps (Protocol 13)

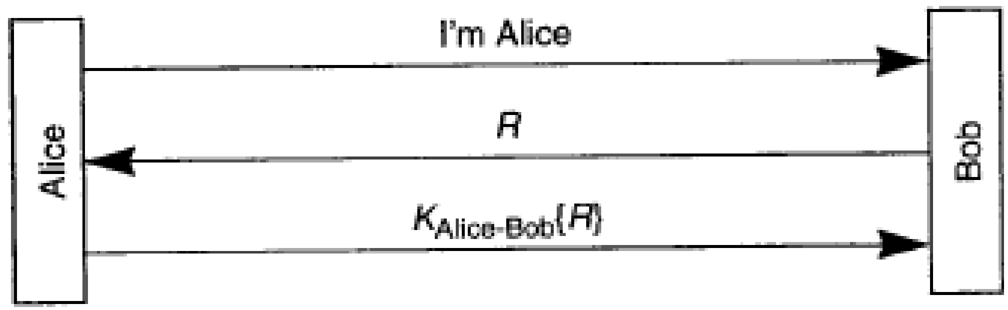


Protocol 11-13. Mutual authentication based on synchronized clocks and a shared secret $K_{\text{Alice-Bob}}$

- Issue: reflection attack, clock synchronization

By Shared Secret Keys

- Alice and Bob has a shared secret K.



Protocol 11-14. Authentication with shared secret

- After authentication, R.
- A good session key
  - Different for each session
  - Unguessable by an eavesdropper
- How about
  - K xor R, $\{K\}_K$, $\{R+1\}_K$?
  - $\{K+R\}_K$, $\{R\}_{R+K}$
  - hash(K|R),

By two-way public keys

- Alice and Bob know their own private keys and know other's public keys.
- Alice picks a random K, sends to Bob $\{K\}_{Bob}$
    - Issue: Trudy can do that too.
- Alice picks a number K, sends to Bob $[\{K\}_{Bob}]_{Alice}$
    - Issue: if Trudy overruns Bob, be able to decrypt recorded conversation.
- Alice sends Bob $[\{K_1\}_{Bob}]_{Alice}$ and Bob sends Alice $[\{K_2\}_{Alice}]_{Bob}$, $K=K_1$ xor K_2.

- Diffie-Hellman key establishment
    - Forward secrecy: even if Trudy overruns both Alice and Bob, she won't be able to decrypted old (recorded) conversation.

By one-way public key

- Only Bob has public/private key pair.
- Alice picks K, sends Bob $\{K\}_{Bob}$
  - No forward secrecy when Bob is overran.
- Diffie-Hellman exchange.