IT-Security 1

Chapter 6: Authentication and Key Agreement

Prof. Dr.-Ing. Ulrike Meyer

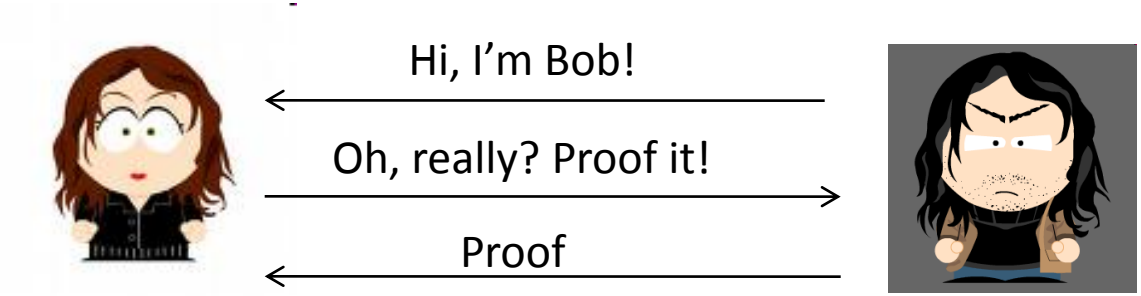
WS 15/16

Chapter Overview

- Authentication
  - Based on Passwords
  - Based on secret keys in general
  - Based on asymmetric keys
- Key Agreement
  - Key transport
  - Key establishment
    - Based on secret keys
    - Based on private/public keys

Definition

- Entity authentication is the process whereby one party is assured
  - of the identity of a second party involved in a protocol
  - and that the second has actually participated
    - i.e., is active at, or immediately prior to, the time the evidence is acquired
- One speaks of mutual authentication if both participating parties authenticate each other



Objectives of Authentication Protocols

- Correctness: In the case of honest parties A and B, A is able to successfully authenticate itself to B, i.e., B has accepted A's identity
- Resistance against transferability: B cannot reuse an identification exchange with A in order to successfully impersonate A to a third party C
- Resistance against impersonation: The probability is negligible that any party C distinct from A, carrying out the protocol and playing the role of A, can cause B to complete the protocol and accept A's identity
- The previous points remain true even if: a large number of previous authentications between A and B have been observed

Many Ways to Prove Who You Are

- What you know
  - Passwords and PINs
  - Secret key
- Where you are
  - IP address, GPS coordinates
- What you are
  - Biometrics
- What you have
  - Secure tokens
  - Secret / private key

Other Aspects

- Usability
  - Hard-to-remember passwords?
  - Carry a physical object all the time?
  - Number of passwords constantly increases
- Denial of service
  - Stolen wallet
  - Attacker tries to authenticate as you $\Rightarrow$ account locked after three failures
  - “Suspicious” credit card usage
- Social engineering
  - Talk someone into giving you his passwords, key or token

Password-Based Authentication

- User has a secret password and System checks it to authenticate the user
- How is the password communicated to the system?
  - Eavesdropping risk
- How can the password be remembered?
- How is the password stored on the system?
  - In the clear? Encrypted? Hashed?
- How does the system check the password?
- How easy is it to guess the password?
  - Easy-to-remember often = easy to guess
  - Password file is difficult to keep secret

Passwords and Computer Security

- First step after any successful intrusion: install sniffer or key logger to steal more passwords
- Second step: run cracking tools on password files
  - E.g. on another hijacked computer
- In Mitnick's "Art of Intrusion", 8 out of 9 exploits involve password stealing and/or cracking

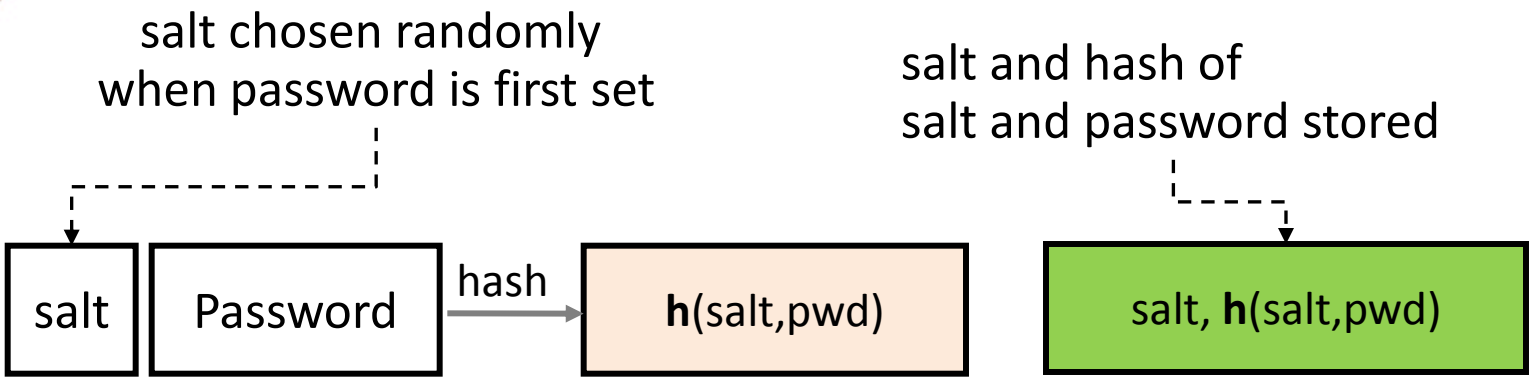
Password Hashing

- Instead of storing passwords in the clear, typically a hash of the password is stored
- When user enters password, the system computes the hash of the password and compares it with the entry in the password file
  - System does not store actual passwords
  - Difficult to go from hash to password
    - Do you see why hashing is better than encryption here?
- Hash function **h** has to be pre-image resistant
  - Given **h**(password) it is hard to find the password

Dictionary Attacks

- A dictionary attack is the guessing of a password by repeated trial and error where the guesses are taken from a “dictionary”
- Dictionary attack is possible because many passwords come from a small dictionary
  - Attacker can pre-compute **h**(word) for every word in the dictionary
    - Pre-computation needs to be done only once!!
    - No access to a password file or the authentication function necessary during pre-computation
- Two types of dictionary attacks:
  - Offline: Password file with hashed passwords in possession of attacker
    - Once password file is obtained, cracking is instantaneous
  - Online: Only the authentication function is available but not password file
    - As e.g. the case for web email accounts
- Note: attacks can be both, against a particular user or against any user!

Salting to Slow Down Dictionary Attacks



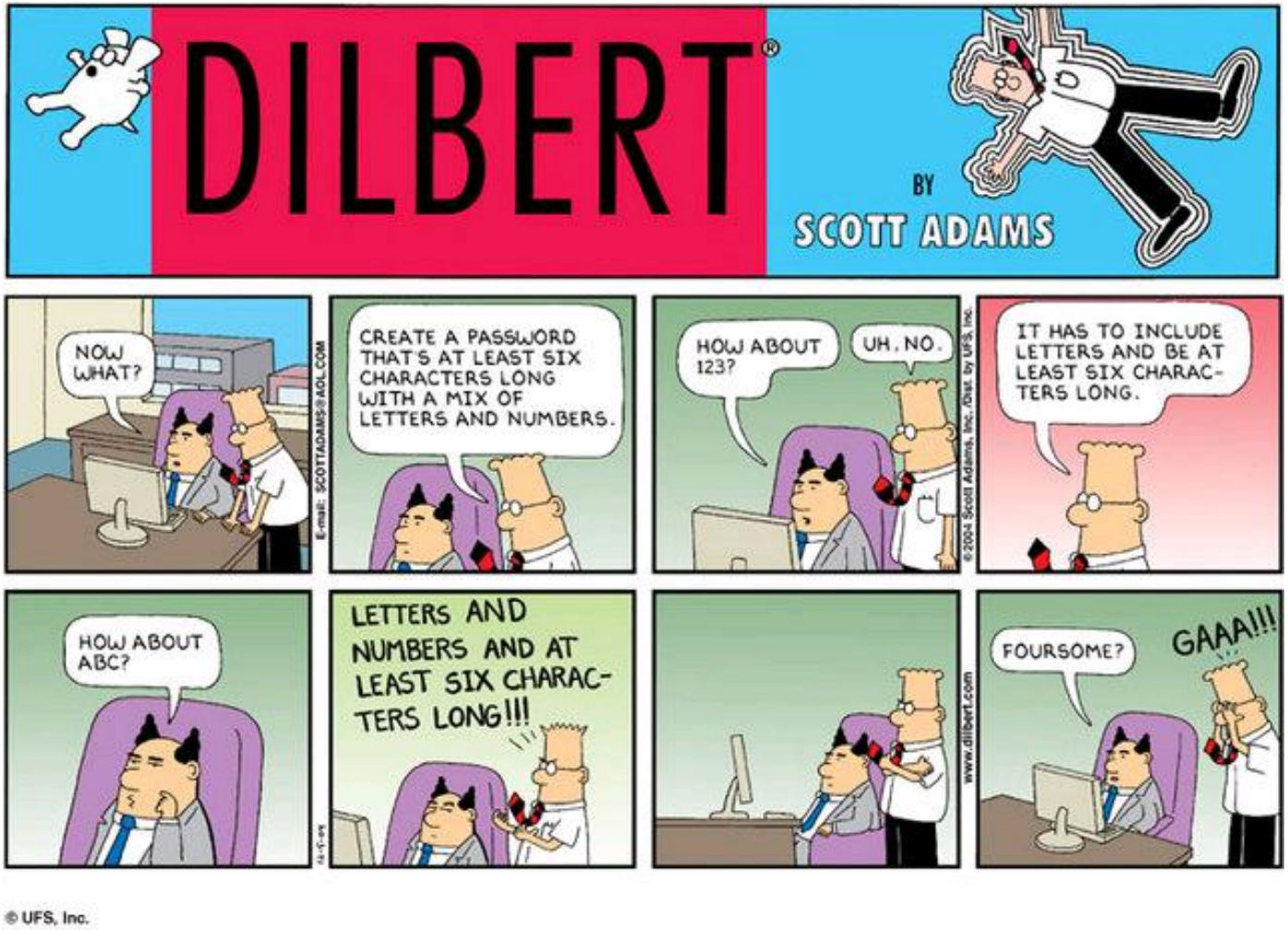
- Instead of hashing the password only, the password and a randomly picked “salt” value are hashed
- For each user the salt and the hash of salt and password is stored in a password file
- Users with the same password have different entries in the password file
- Offline dictionary attack becomes much harder

Advantages of Salting

- Without salt, attacker can pre-compute hashes of all dictionary words once for all password entries
  - Same hash function on all UNIX machines; identical passwords hash to identical values
  - One table of hash values works for all password files
- With salt, attacker must compute hashes of all dictionary words once for each combination of salt value and password
  - With 12-bit random salt, same password can hash to 4096 different hash values

Example: UNIX Password System

- Originally, DES encryption used as if it was a hash function
  - Encrypt NULL string using password as the key
    - Truncates passwords to 8 characters!
  - Can instruct modern UNIXes to use any cryptographic hash function
- Problem: passwords are not truly random
  - With 52 upper- and lower-case letters, 10 digits and 32 punctuation symbols, there are $94^8 \approx$ 6 quadrillion possible 8-character passwords
  - However, humans like to use dictionary words, human and pet names $\approx$ 1 million common passwords


DILBERT®
BY
SCOTT ADAMS
NOW WHAT?
CREATE A PASSWORD THAT'S AT LEAST SIX CHARACTERS LONG WITH A MIX OF LETTERS AND NUMBERS.
HOW ABOUT 123?
UH, NO.
IT HAS TO INCLUDE LETTERS AND BE AT LEAST SIX CHARACTERS LONG.
HOW ABOUT ABC?
LETTERS AND NUMBERS AND AT LEAST SIX CHARACTERS LONG!!!
FOURSOME?
GAAA!!!
E-mail: SCOTTADAMS@AOL.COM
© 2004 Scott Adams, Inc./Dist. by UFS, Inc.
www.dilbert.com

Password Selection (1)

- User selected?
  - Users tend to select dictionary words or slight ramifications that can still be broken with the help of a dictionary attack
- Typical user habits include
  - Passwords based on account names
    - E.g. account name followed by a number
  - Passwords based on user names
  - Name reversed, all letters upper case, all letters lower case
  - First initial followed by last name reversed
  - Passwords based on computer names
  - (Reversed) dictionary words
  - (Reversed) dictionary words with some or all letters capitalized

Password Selection (2)

- Dictionary words with arbitrary letters turned into control characters
- Conjugations or declensions of words
- Patterns from the keyboard
- Passwords shorter then 6 characters
- Passwords containing only digits
- Passwords that look like a license plate number
- Acronyms
- Concatenations of dictionary words
- Dictionary words proceeded or followed by punctuation marks and/or digits
- Dictionary words with all vowels deleted
- Passwords with too many characters in common with the last used password

Password Selection (3)

- Randomly?
  - Random passwords cannot be remembered by users
    - Psychological studies show that users can only remember passwords of up to 8 characters and on average only one of them
    - Consequence: users have to write down passwords
  - Randomly selected passwords have to exclude dictionary words and their ramifications
  - Random number generator has to be strong

Password Selection (4)

- Pronounceable but non-dictionary?
  - Compromise between random passwords that are not memorizeable and dictionary words
  - Pronounceable passwords are the ones that make use of the pronounceable phonemes in a language
    - E.g. in English “helgoret” and “juttelon” are pronounceable while “wrzpft” is not
  - Requires longer passwords
    - Possible solution is to allow users to type in longer passwords but hash them to a smaller size before using them in the system

Password Selection (5)

- Proactive password checking
  - Let user select the password
  - Allow system to reject the password based on same rational under which dictionaries for dictionary attacks are constructed
- Proactive password checking should
  - Be mandatory
  - Reject any easy guessable password
  - Discriminate on a per user basis
  - Discriminate on a per site basis
  - …

Guessing Through Authentication Function

- Password file and hash function information not available
  - Try to guess password by entering guess into the authentication function
- Cannot be prevented as the authentication function needs to be available to any user (e.g. web login to email accounts)
- Defense options
  - Exponential backoff
    - first entry of wrong password: wait x^0 seconds before accepting second entry
    - Second entry of wrong password: wait x^1 seconds…
  - Disable after n wrong entries
  - Disconnect (works only if connection establishment time intensive)

Other Password Security Risks

- Keystroke loggers
  - Hardware
    - KeyGhost, KeyShark, others
  - Software (spyware)
  - Shoulder surfing
- Same password at multiple sites
- Broken implementations
- Social engineering

Default Passwords

- Examples from Mitnick's "Art of Intrusion"
  - U.S. District Courthouse server: "public" / "public"
  - NY Times employee database: pwd = last 4 SSN* digits
  - "Dixie bank": break into router (pwd="administrator"), then into IBM AS/400 server (pwd="administrator"), install keylogger to snarf other passwords
    - "99% of people there used 'password123' as their password"

- *: SSN = Social Security Number

Social Engineering

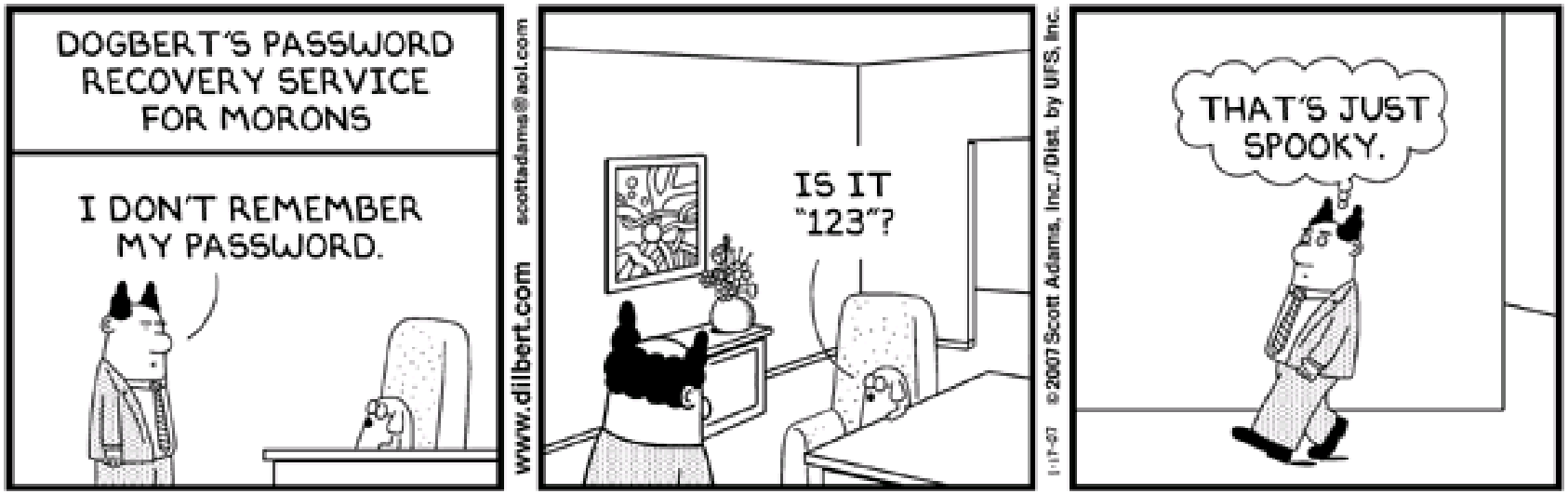
- Univ. of Sydney study (1996)
  - 336 CS students emailed asking for their passwords
    - Pretext: “validate” password database after suspected break-in
  - 138 returned their passwords; 30 returned invalid passwords; 200 reset passwords (not disjoint)

- Treasury Dept. report (2005)
  - Auditors pose as IT personnel attempting to correct a “network problem”
  - 35 (of 100) IRS managers and employees provide their usernames and change passwords to a known value

Strengthening Passwords

- Add biometrics
  - For example, keystroke dynamics or voiceprint
  - Revocation is often a problem with biometrics
- Graphical passwords
  - Goal: increase the size of memorable password space
- Rely on the difficulty of computer vision
  - Face recognition is easy for humans, hard for machines
  - E.g. present user with a sequence of faces, he must pick the right faces several times in a row to log in

Password Recovery



Password Reset Mechanisms (1)

- Users often forget their password such that the systems need to support password resets
  - E.g. 4,28 % of users of yahoo.com forget their password in a three month period
- Common Password reset mechanisms include
  - Security questions
    - What's your favorite actor?
  - Use of another channel
    - Assign a new temporary password over another channel e.g. via SMS, voice calls, letters, (secondary) email address,…
    - Ability to receive message over this other channel used as proof of identity

Password Reset Mechanisms (2)

- Require user to show up in person
- Additional long term-key
  - E.g. PUK = Personal Unblocking Key for mobile phones
- Email with clickable link to reset page
- Combination of security questions and email
  - E.g. require the user to answer security questions before sending an email

Security Questions (1)

- Questions, for which users can easily remember the answers as they are part of their long-term memory
- Are e.g. used by many US financial institutions
- When creating an account the user selects one or more challenge questions and provides the answers to them
- If a user claims to have forgotten his password he is presented with these questions
- Entering the answer(s) leads him to the page where he can reset his password

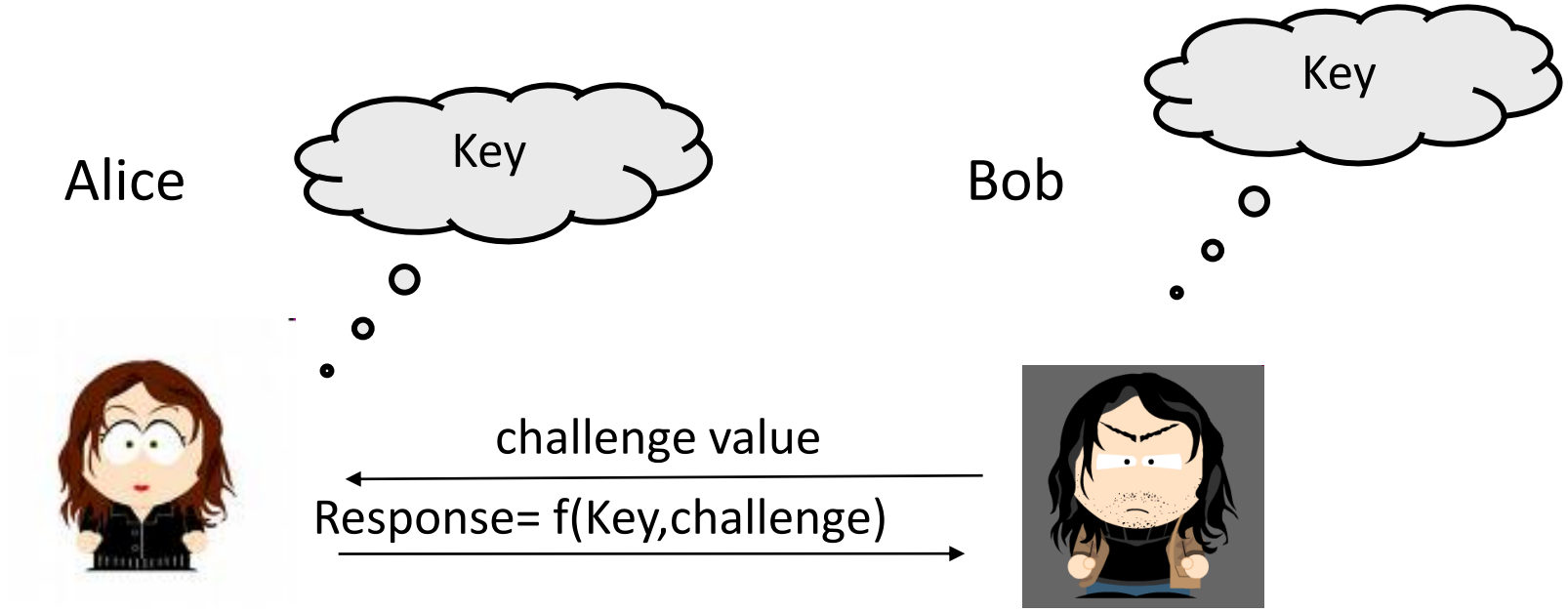
Challenge Questions (2)

- Questions can be classified into
  - Personal questions
    - Ask for personal information such as mother's maiden name,..
    - Problem: Answers are often easy to obtain from personal information available on the web today (Facebook, Xing, etc.)
    - Problem: Personal acquaintances can typically also answer these question
    - Problem: Similar questions used everywhere
  - Sensitive questions
    - Ask for sensitive information such as the ATM PIN, the social security number, etc.
    - Less easy to guess, although e.g. social security number may be in possession of an attacker

Example: Gmail

- Requires you to type in your account name
- Presents you with a visual challenge which you have to type in
- Sends you a clickable link to your secondary email address
- If you do not have a secondary email address you have to wait for 5 days and retry again
- Only if your account was inactive during this 5 day period you are presented with the password security questions you entered when you created the account

Challenge-Response Authentication



- Challenge values are typically random numbers
- Responses depend on a shared secret key, a password or a private key corresponding to a public key

Challenge-Response Authentication

- In challenge-response authentication protocols
  - Claimant proofs its possession of a secret to the verifier without sending it
- The challenge is typically a random number or a time stamp
- The response is a key-dependent function of the challenge
- Challenge-response protocols can be based on
  - Symmetric encryption algorithms
  - Keyed hash functions
  - Digital signatures

Example: Lamport's One-Time Passwords

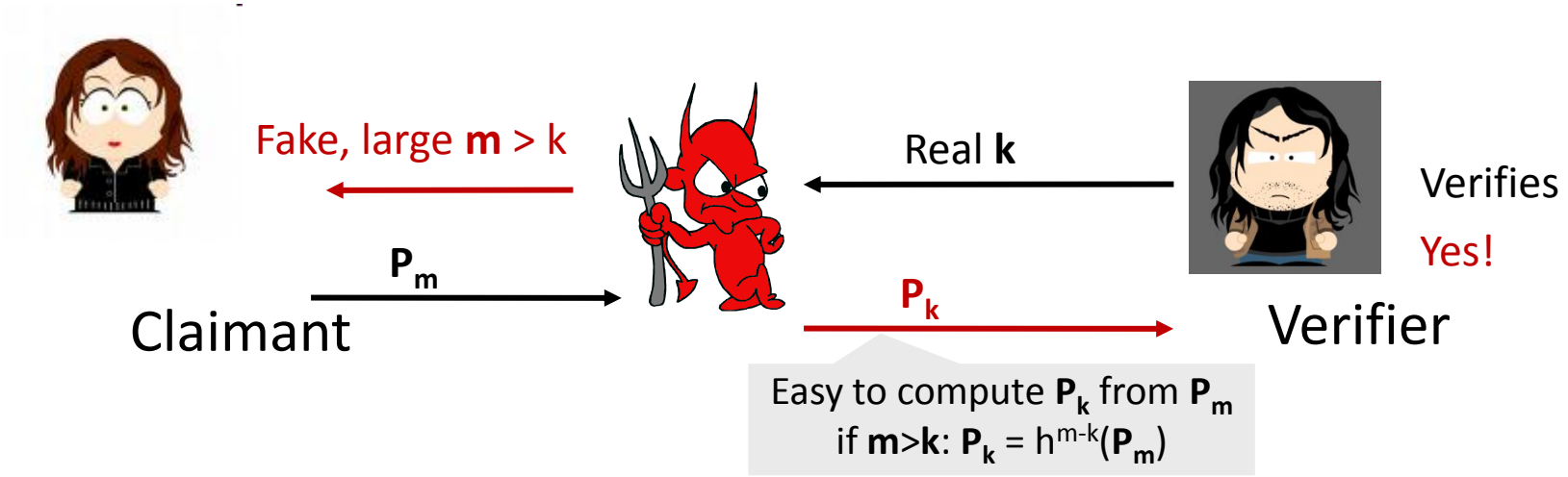
- Let h be a one-way hash function
- Then the claimant chooses an initial seed S and computes h(S) = S_1, $h(S_1) = h^2(S) = S_2$, …, $h(S_{n-1}) = h^n(S) = S_n$
- The claimant provides $P_0 = S_n$ to the verifier
- The passwords in order of usage then are
  - $P_1 = S_{n-1}$, $P_2 = S_{n-2}$,…

Challenge: **k**

Response: $\mathbf{P_k}$

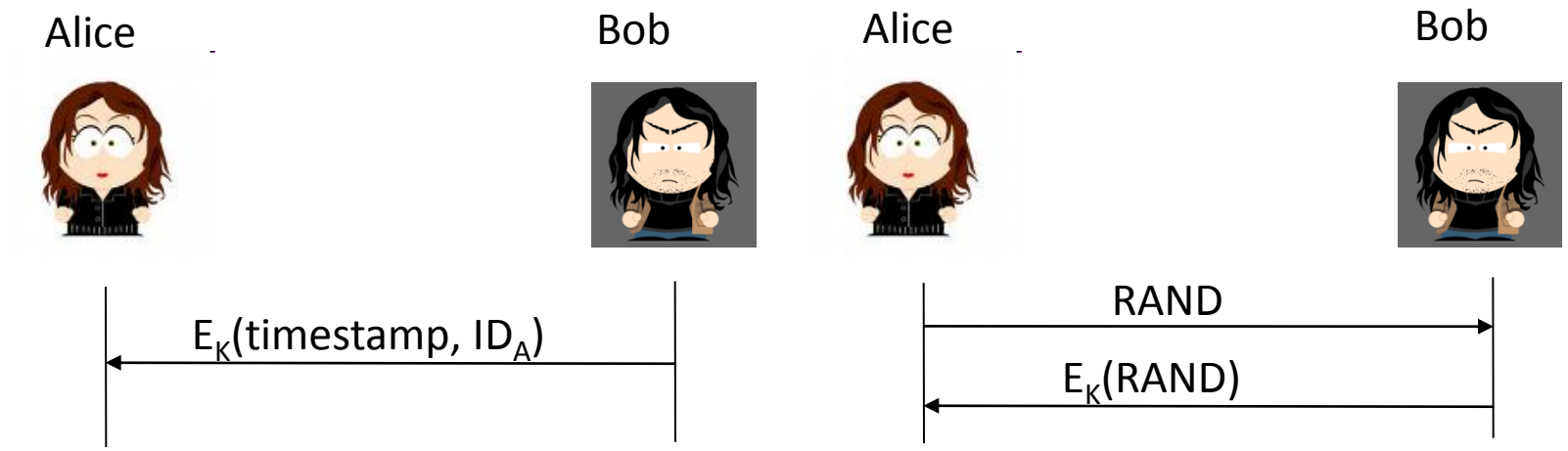
Checks if $h(\mathbf{P_k}) = \mathbf{P_{k-1}}$

“Small k” Attack



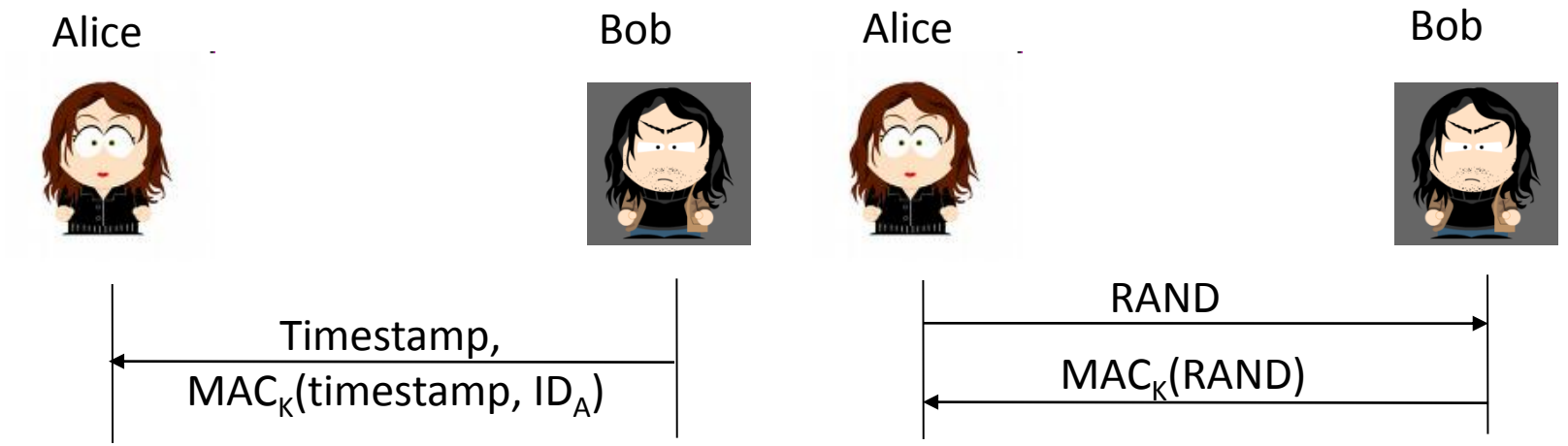
- First message from Bob is not authenticated!
- Alice should remember current value of k
- Still an attacker may be able to send a fake challenge, obtain a valid password and use it to impersonate the claimant to the verifier later on

Building Blocks for Unilateral Authentication (1)



- Based on timestamps (left) or random numbers (right) and symmetric encryption
- Identifier of verifier included (left) to avoid that attacker immediately reflects the authentication to the claimant
- Both protocols assume that verifier and claimant already obtained knowledge of their claimed identifiers

Building Blocks for Unilateral Authentication (2)

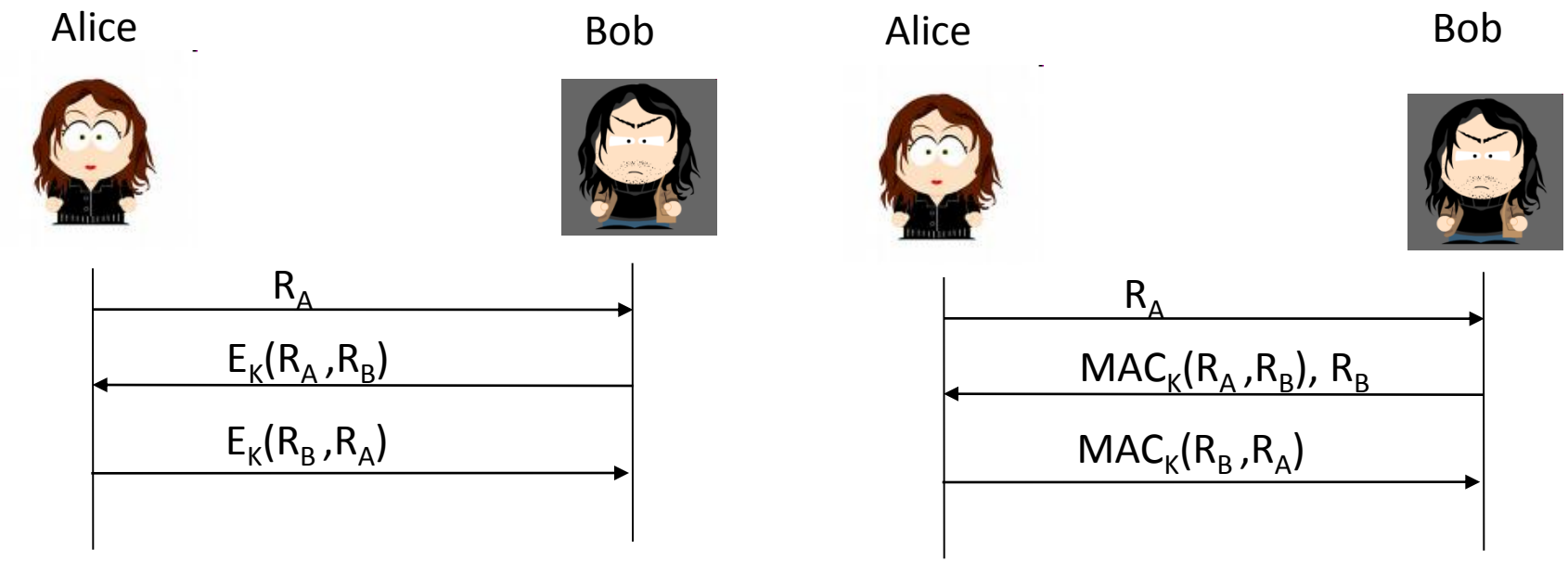


- Here, a message authentication code is used instead of symmetric encryption

Properties of The Four Protocols Building Blocks Above

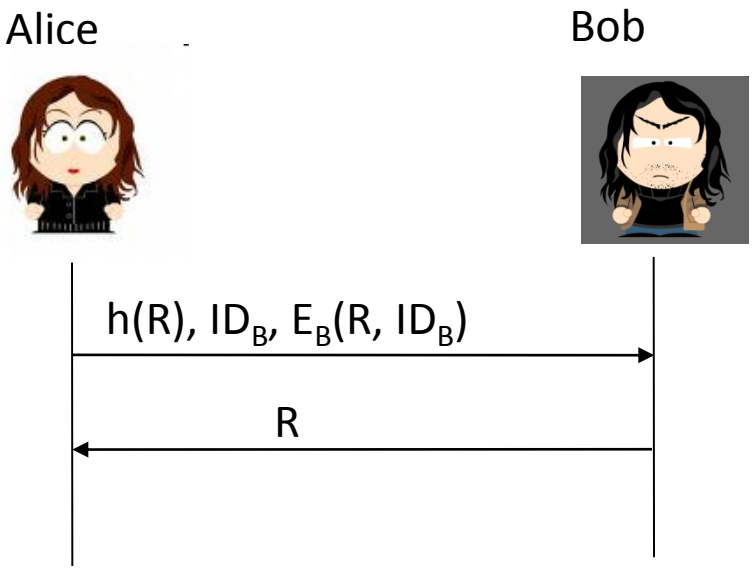
- In all four protocol building blocks
  - Bob proofs its identity to Alice by possession of a shared secret key
  - Only Bob (and Alice) can compute valid responses
- The timestamp and the RAND respectively guarantee that Bob participated in the protocol, i.e. that no one can replay Bobs response without Alice's notice
- Note that Alice cannot impersonate Bob to a third party as Bob shares a different key with that third party

Building Blocks for Mutual Authentication



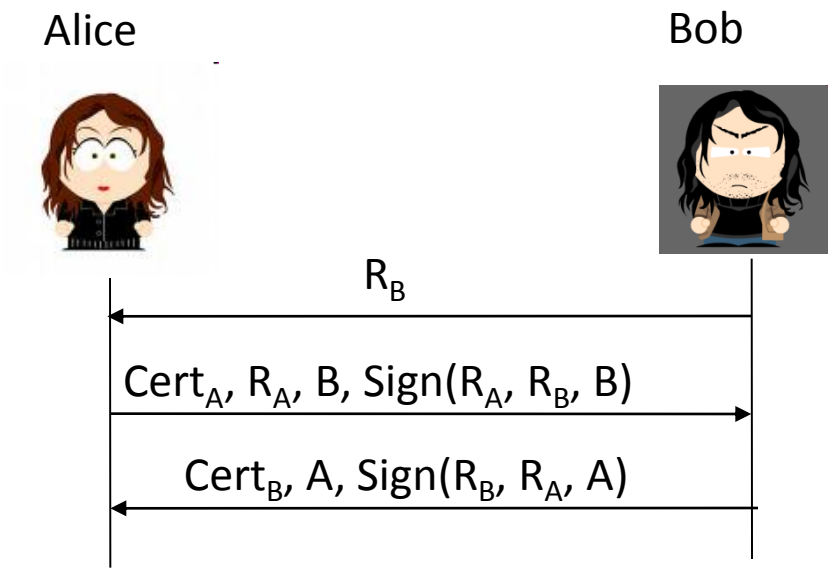
- Second random number R_B serves as challenge from Bob to Alice
- Encrypting/computing a message authentication code on R_B together with R_A additionally prevents chosen plaintext (chosen message) attacks

Using Public Key Encryption



- Provides unilateral authentication
- h(R) proofs to claimant that verifier knows R and is not just trying to fool claimant into decrypting R
- R and ID_B are encrypted by the verifier with the claimants public key
- Mutual authentication can be achieved similarly

Using Digital Signatures



- Example for a building block for mutual authentication using digital signatures

Use of One-Time Passwords

- Authentication of a user to a system often uses RSA SecureID
- Authentication through a PIN combined with the possession of a SecureID token
- System and token are time-synchronized
- Every 60 seconds a new one-time-password is generated on the token
- When authenticating to the system the user enters its username, the PIN and the current one-time password on the display of his token
- The one-time password is generated from a 64-Bit-Seed that is generated from the serial number and the current time with the help of AES

Authentication Alone…

- … is not sufficient
- All of the protocols discussed so far are useless if they are not combined with a key establishment
- Only the use of the key for integrity protection will then guarantee that the same two parties that authenticated each other are still communicating

Key Establishment Protocols

- Protocol that establishes a shared secret key between two parties
  - Shared secret key then used as session key or used to derive session keys
- Can be divided into

  - Key transport protocols
    - Key generated by one party or a trusted third party and securely "transported" to the designated endpoints

  - Key agreement protocols
    - a shared secret key is derived by two parties as a function of information contributed by each of these (ideally)

- Example for key agreement protocol: Diffie-Hellman

Motivation for the Use of Session Keys

- Limit available ciphertext (under a fixed key) for cryptanalytic attack
- Limit exposure, with respect to both time period and quantity of data, in the event of (session) key compromise
- Avoid long-term storage of a large number of distinct secret keys
  - in the case where one terminal communicates with a large number of others by creating keys only when actually required;
- Create independence across communications sessions or applications.

Potential Characteristics of Key Establishment

- Key freshness – A key is fresh (from the viewpoint of one party) if it can be guaranteed to be new, as opposed to possibly an old key being reused
- Key control - In some protocols (key transport), one party chooses a key value. In others (key agreement), the key is derived from joint information, and neither party is able to control or predict the key
- Efficiency - Considerations include:
  - number of message exchanges (passes) between parties
  - bandwidth required by messages (total number of bits transmitted)
  - complexity of computations by each party (as it affects execution time)
  - possibility of pre-computation to reduce on-line computational complexity
- Third party requirements – include:
  - requirement of an on-line (real-time), off-line, or no third party;
  - degree of trust required in a third party (e.g., trusted to certify public keys vs. trusted not to disclose long-term secret keys

Perfect Forward Secrecy, Known Key Attacks

- When analyzing security of key establishment protocol
  - Impact of compromise of various types of keying material should be considered, in particular
    - Compromise of long-term secret keys
    - Compromise of past session keys

- A protocol is said to have perfect forward secrecy if a future compromise of long-term keys does not compromise past session keys
  - Example: Authenticated Diffie-Hellman key agreement. Even if long-term signature keys are compromised, past session keys can still not be recovered

- A protocol is said to be vulnerable to a known-key attack if compromise of past session keys either allows
  - a passive adversary to compromise future session keys
  - or impersonation by an active adversary in the future

Properties of Key Establishment Protocols

- Implicit key authentication
  - Property whereby one party is assured that no other party aside from a specifically authenticated second party (and possibly additional authenticated trusted parties) may gain access to a particular secret key
- Key confirmation
  - Property whereby one party is assured that a second (possibly unauthenticated) party actually has possession of a particular secret key
- Explicit key authentication: implicit key authentication + key confirmation
- Authenticated key establishment protocol
  - key establishment protocol that provides implicit key authentication

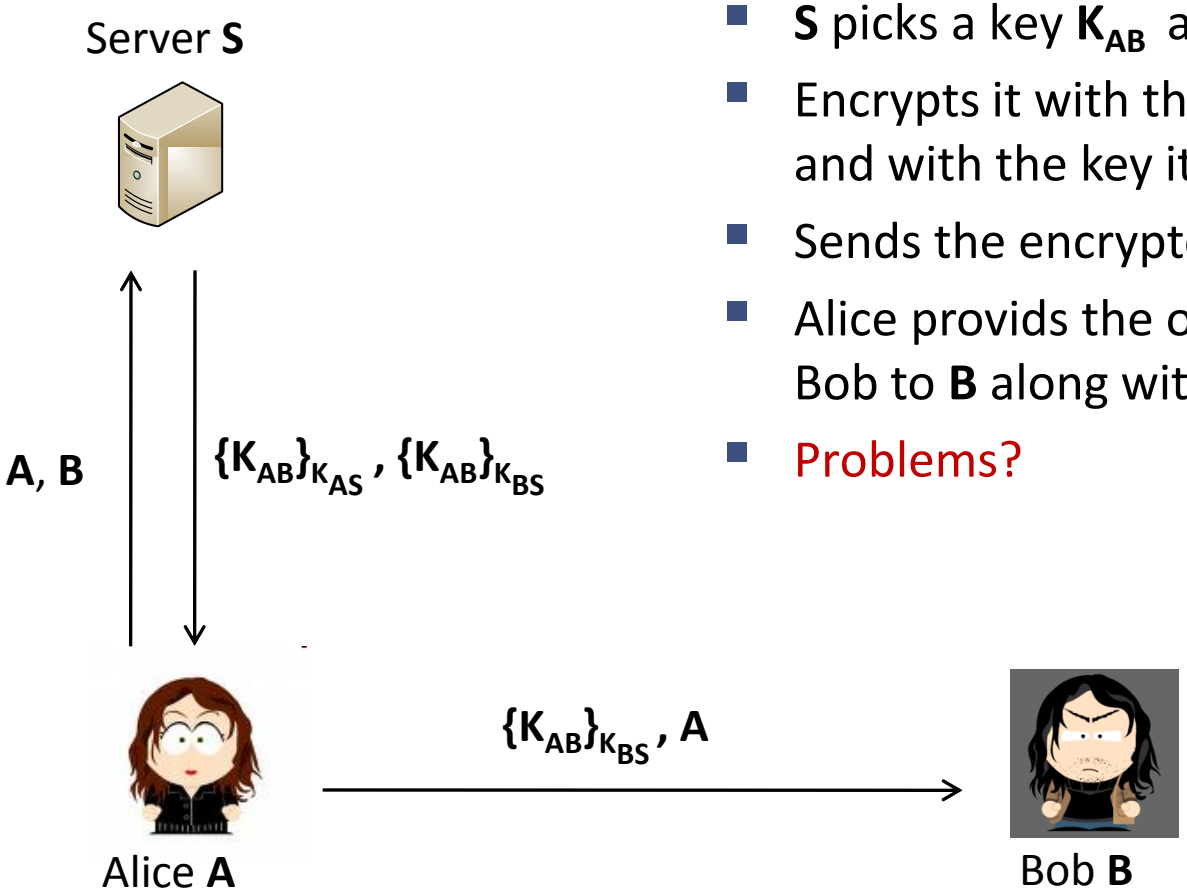
Key Transport

- Key transport using symmetric encryption
  - **A** and **B** share a secret master key **K**
  - **A** chooses session key **SK**, encrypts it with the help of **K** and sends $E_K(\mathbf{SK})$ to **B**
- If additionally key freshness from both parties respectively is needed, nonces can be included
  - Again **A** and **B** share a secret master key **K**
  - **B** chooses a nonce and sends it to **A**
  - **A** chooses a nonce and a session key **SK** and encrypts both nonces and **SK** using **K**: $E_K(\mathbf{SK}, \mathbf{N_A}, \mathbf{N_B})$
- Do these two mechanisms provide perfect forward secrecy?

Use of Trusted Third Parties

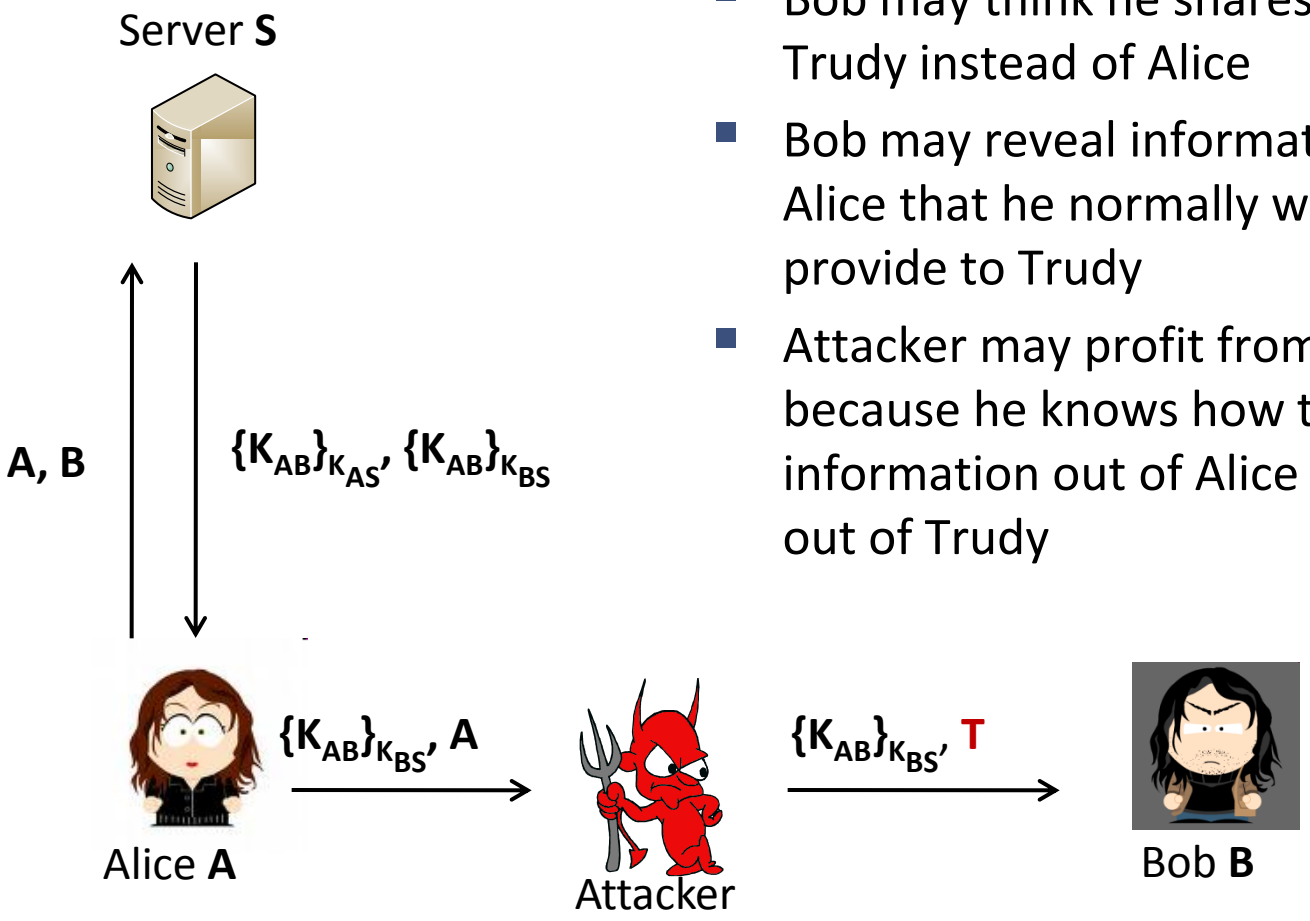
- Many key establishment mechanisms make use of a trusted third party
- The two parties that want to establish a key are already sharing credentials (certificates or secret keys) with the trusted third party and make use of these to establish a secret key between each other
- Depending on the application, the entity playing that role may have changing names
  - Trusted Server
  - Authentication Server
  - Key Distribution Center (KDC)
  - Key Translation Center (KTC)
  - Certification Authority (CA)

Building a Key Establishment Protocol – First Attempt



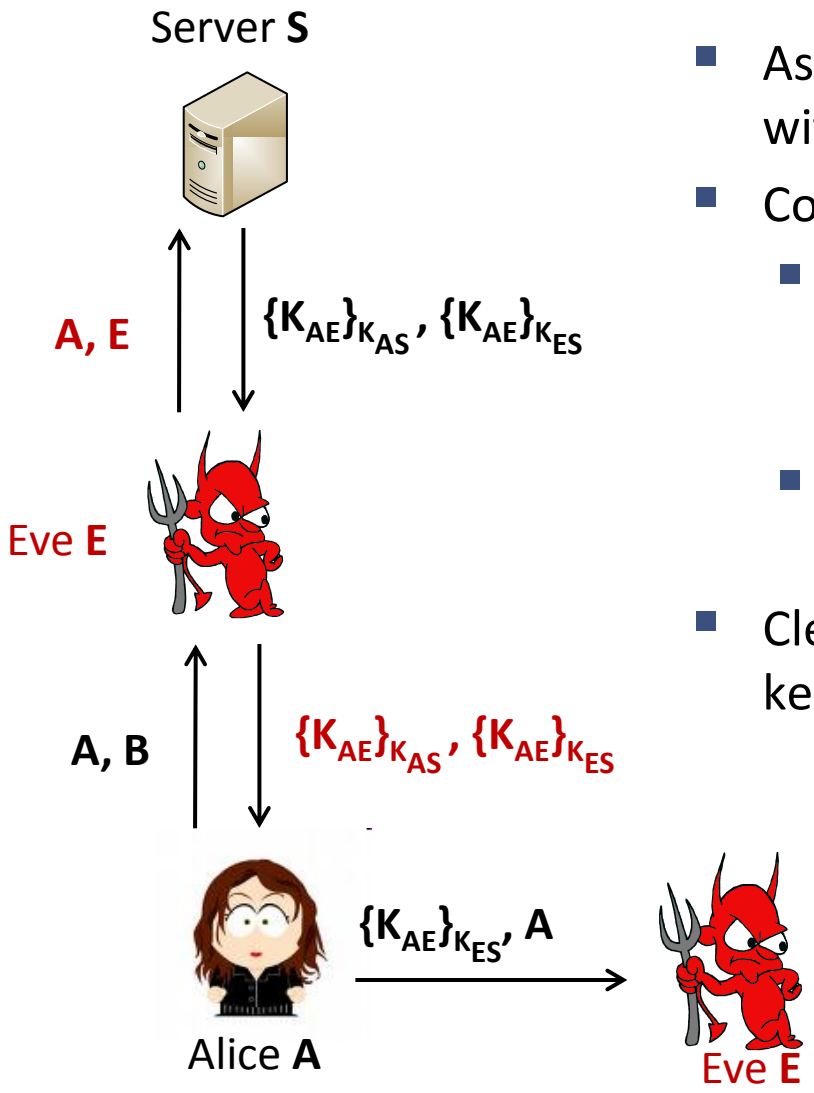
- **A** requests key for **A** and **B** from **S**
- **S** picks a key K_{AB} at random
- Encrypts it with the key it shares with **A** and with the key it shares with **B**
- Sends the encrypted keys to Alice
- Alice provids the one encrypted for Bob to **B** along with her identifier
- Problems?

First Attack Possible Against the First Attempt



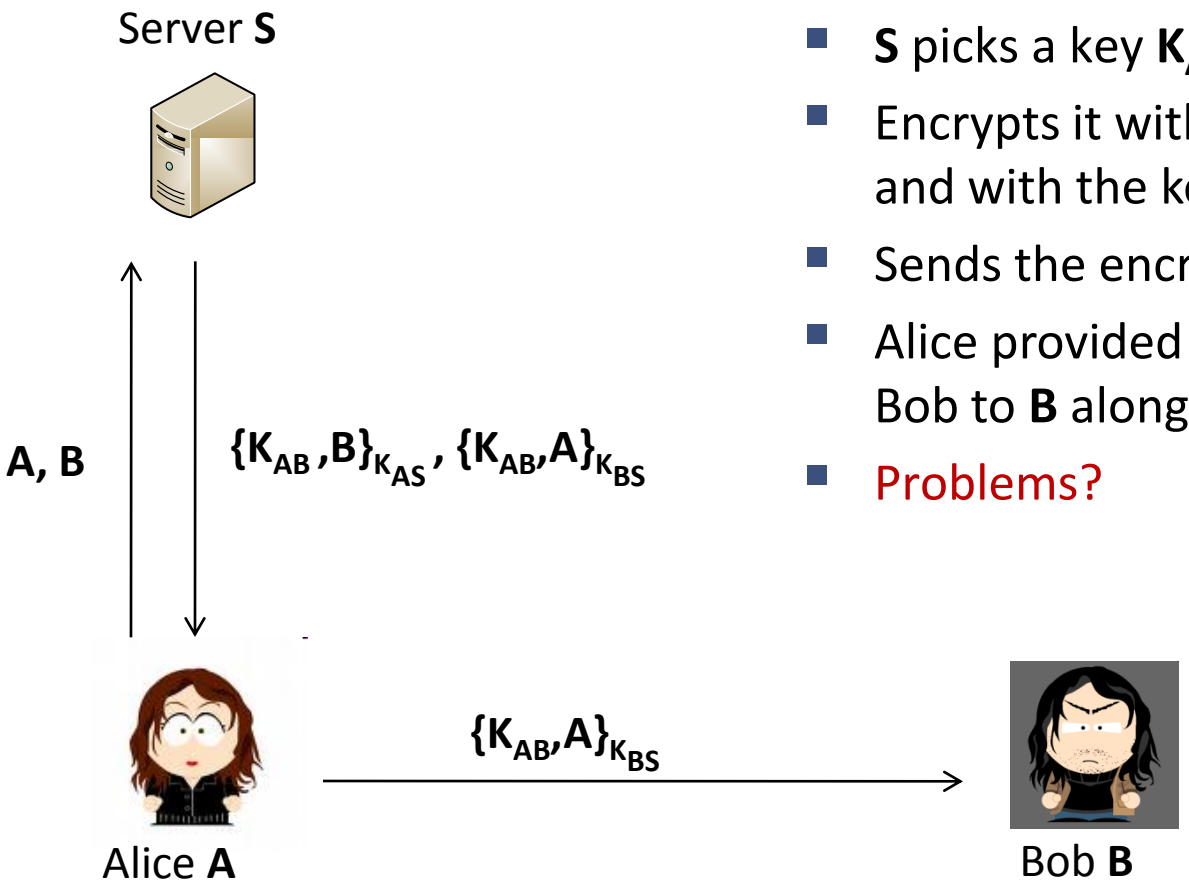
- Consequence
  - Bob may think he shares a key with Trudy instead of Alice
  - Bob may reveal information to Alice that he normally would only provide to Trudy
  - Attacker may profit from this because he knows how to get this information out of Alice but not out of Trudy

Second Attack Possible Against the First Attempt



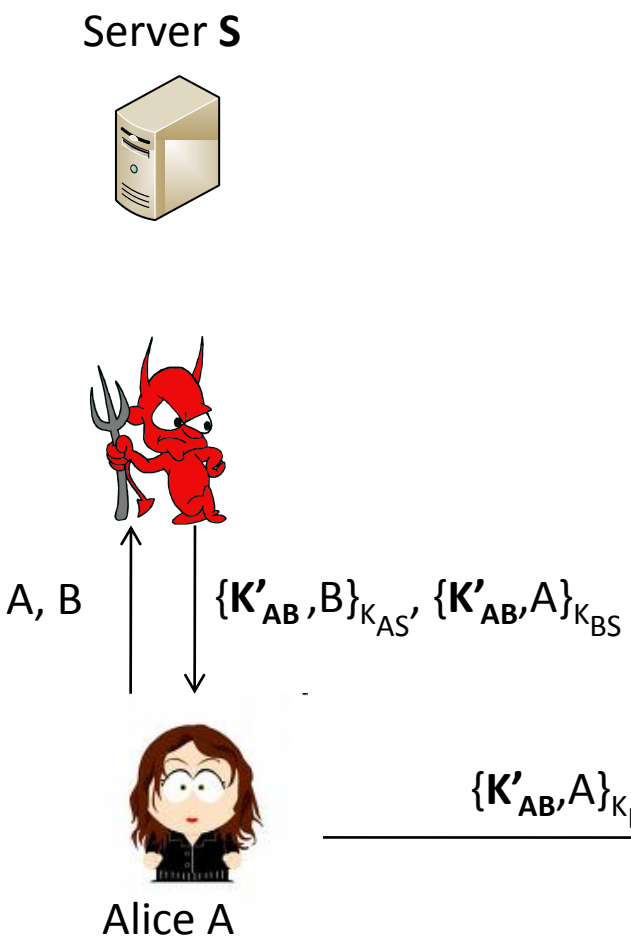
- Assumes attacker also shares a key with the server
- Consequence
  - Attacker **Eve** can make Alice believe that she successfully established a key with Bob
  - Instead the attacker masquerades as Bob
- Clear from this: we need to bind the keys somehow to the identifiers

Building a Key Establishment Protocol – Second Attempt



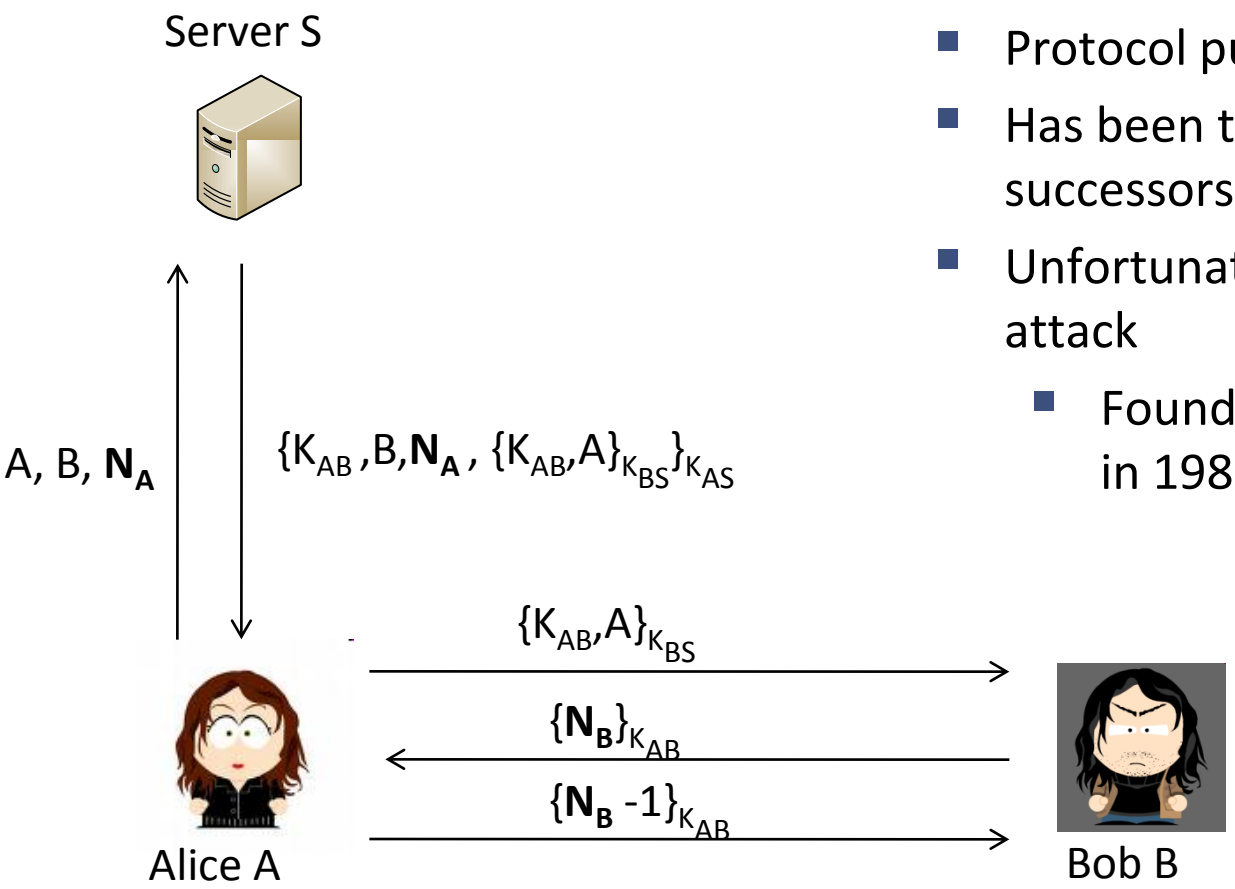
- **A** requests key for **A** and **B** from **S**
- **S** picks a key K_{AB} at random
- Encrypts it with the key it shares with **A** and with the key it shares with **B**
- Sends the encrypted keys to Alice
- Alice provided the one encrypted for Bob to **B** along with her identifier
- Problems?

Attack on the Second Try



- Assume
  - Attacker is able to obtain the value of a past session key K'_{AB}
  - And has recorded the key exchange messages when the old key was established
  - Then the attacker can make Alice and Bob reuse the broken old key again
  - I.e. the protocol is vulnerable to replay attacks

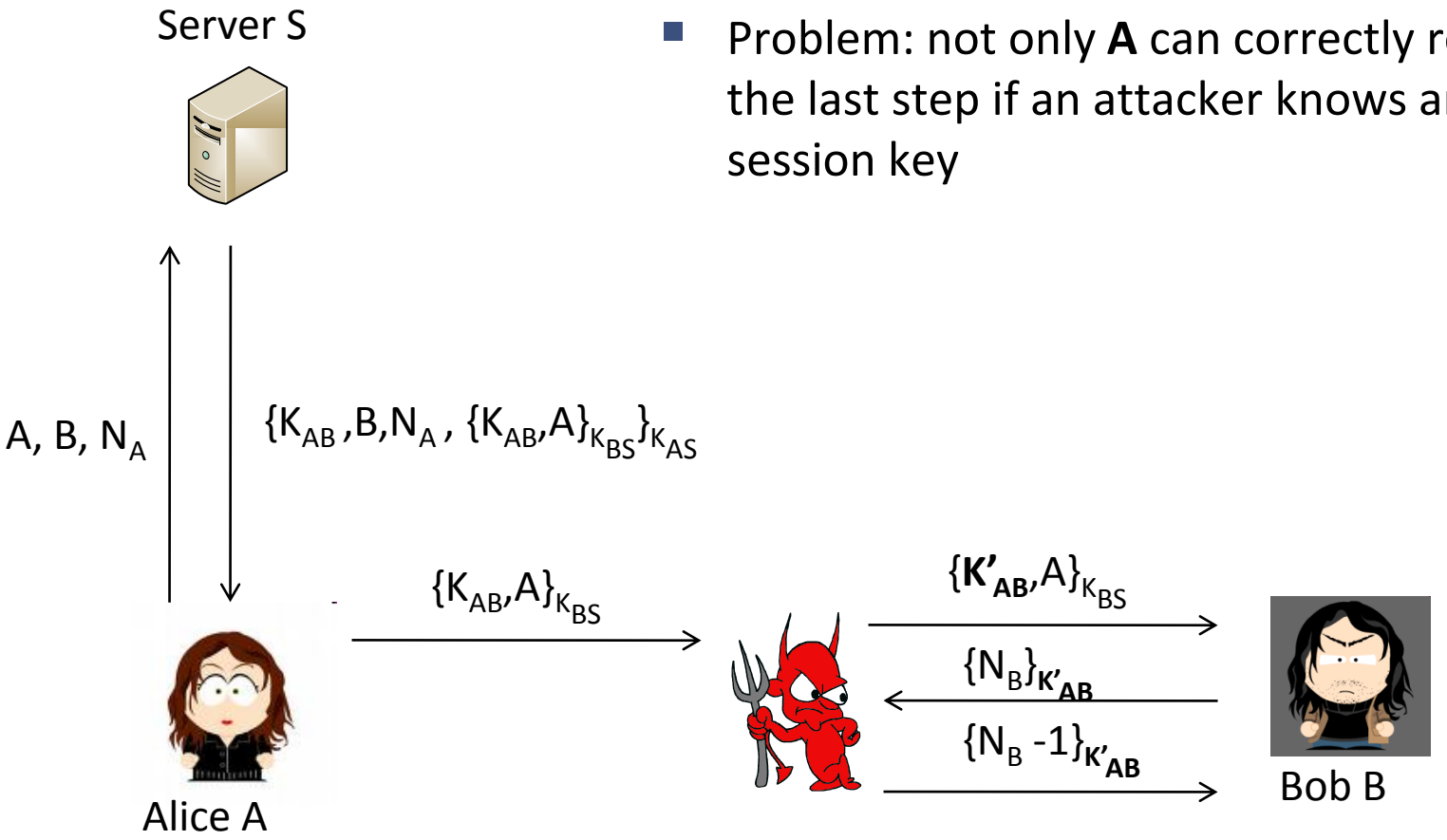
Building a Key Establishment Protocol – Third Attempt also Known as Needham-Schroeder Protocol



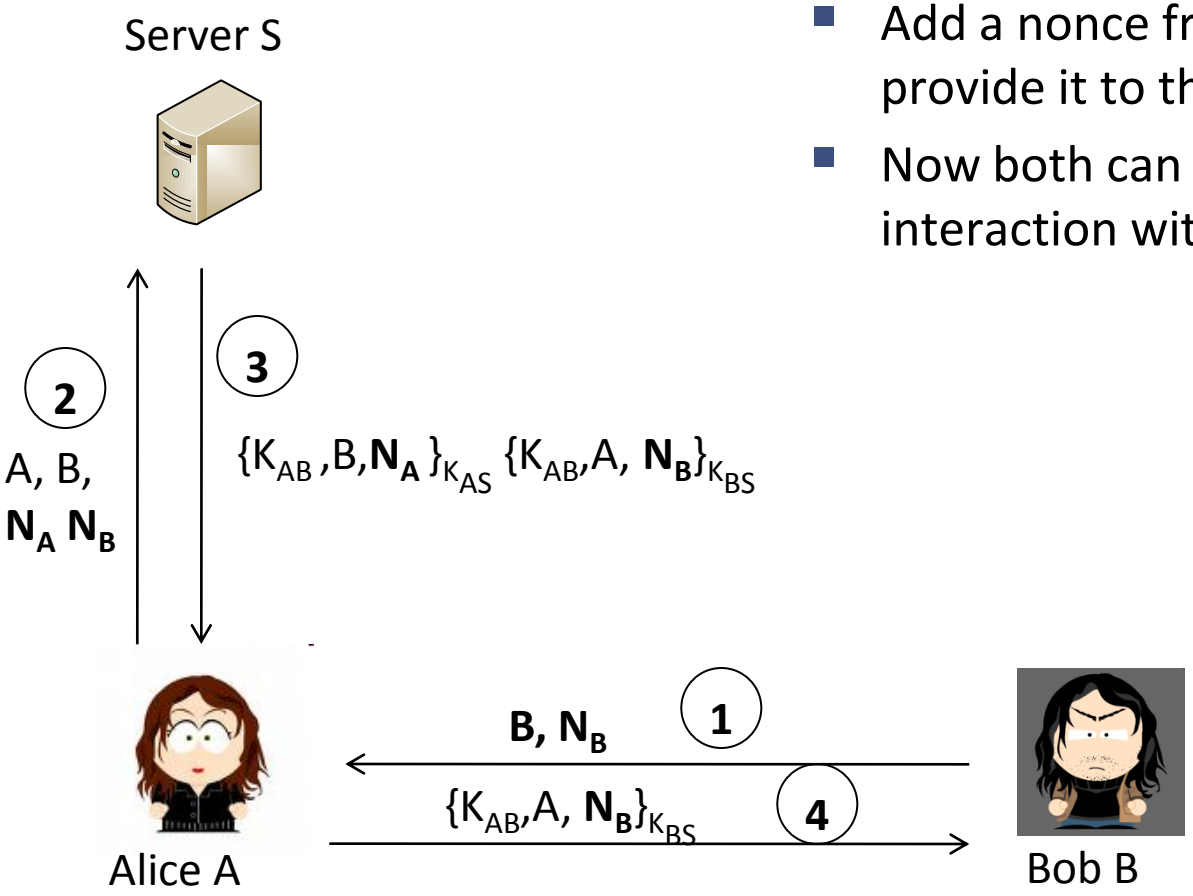
- Add Nonces for replay protection
  - Protocol published in 1978
  - Has been the basis for a lot of successors
  - Unfortunately vulnerable to an attack
    - Found by Denning and Sacco in 1981

Attack Against the Needham Schroeder Protocol

- Found by Denning and Sacco in 1981
- Problem: not only **A** can correctly reply **B** in the last step if an attacker knows an old session key



Building a Key Establishment Protocol – Fourth Attempt



- Add a nonce from each **A** and **B** and provide it to the server
- Now both can check that the interaction with the server is fresh

Key Agreement

- Based on public key cryptography
  - Diffie-Hellman Key Agreement
    - Can be authenticated with help of symmetric keys or public key signatures on the public value (see IKE)
- Based on symmetric keys
  - Authenticated exchange of nonces,
  - Use nonces and long-term shared secret as input to a one-way hash function to generate session keys

References and Further Reading

- Kaufman Chapters 9, 10, and 11
- The basic challenge response protocols are stand
  ISO/IEC 9798
- D. Florencio, C. Herley: “A LargeScale Study of Web Password Habits”, 2007
- A. Rabkin: “Personal knowledge questions for fallback authentication: Security questions in the era of Facebook”, 2008
- Boyd, Mathuria: Protocols for Authentication and Key Establishment, 2003