SYMMETRIC AUTHENTICATION

DR. O. I. ADELAIYE

AUTHENTICATION

- Is a user the person they pretend to be?
- Entails data integrity
  - Does not make sense to be sure about the origin of the message if the message was modified en route !
- Comes before confidentiality and data integrity
- Authorization depends upon authentication
- Three types of authentication
  - Something I am
  - Something I know
  - Something I posses

HIGHLIGHTS

- "Something I possess" based authentication
- "Something I am" based authentication
- "Something I know" based authentication
  - Password-based Authentication
  - Kerberos
- Diffie-Hellman

SOMETHING I POSSESS

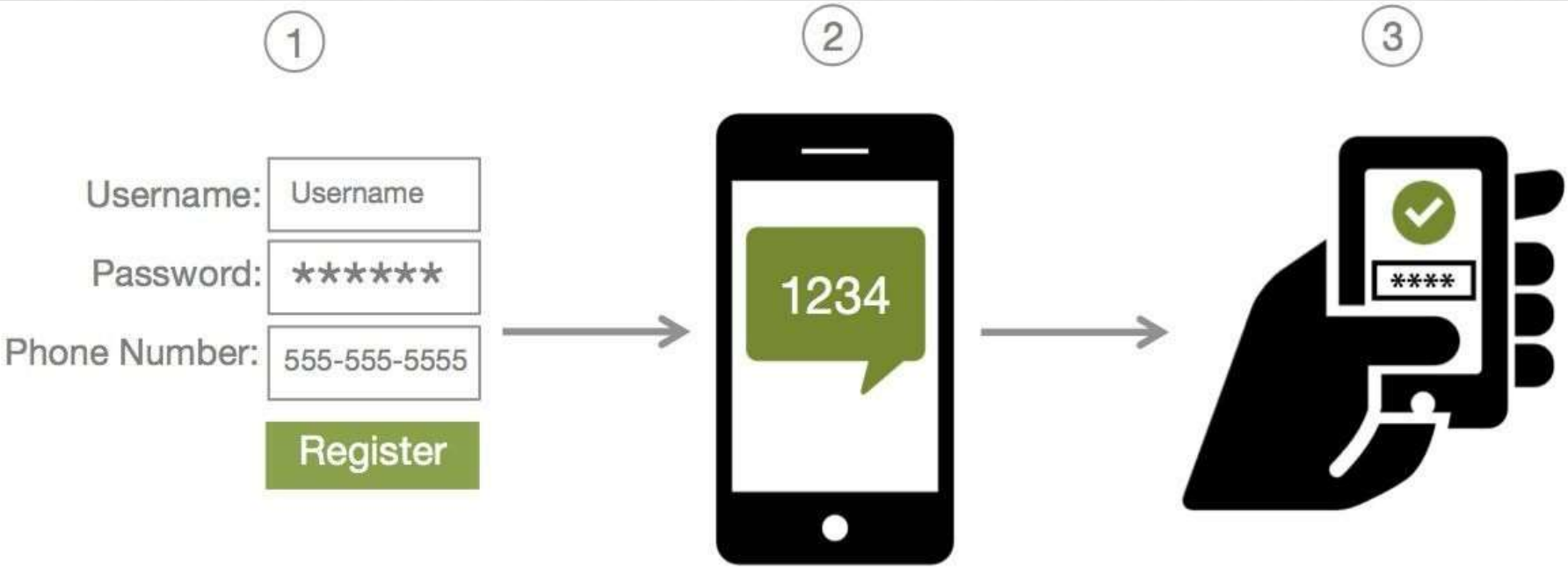
SOMETHING I POSSESS????

- Physical
  - Proximity card
  - Smartcard
  - Visa card
  - Private key
  - Passport
  - Mobile phone
- Attacks are “easy” to detect

SOMETHING I POSSESS????

- These are usually two factor authentication systems, and rely on something you possess and something you know
  - e.g. mobile phones-based authentication

AUTHENTICATION USING MOBILE PHONES



Upon signing up for a new web or mobile app, the user provides their mobile phone number.

With the help of Plivo's SMS API, your app sends the user a unique code via a SMS to their mobile during an event (e.g., new registration, password reset, etc).

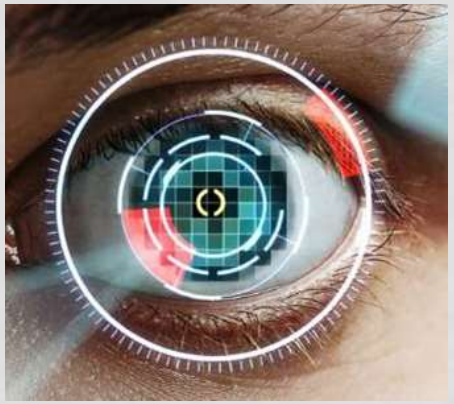
Users then enter the unique code into your app to securely verify their identity and start using the app.

Source: www.pilvo.com

SOMETHING I AM

SOMETHING I AM???

- Fingerprint
- DNA testing
- Retina Scans
- Potentially the strongest type of authentication



SOMETHING I AM???

- But many problems with them today
  - The comparison is an approximate match against a template and this produces False Positives and False Negatives
  - The biometric needs to be captured live since they are not secret and can be copied/stolen e.g. a fingerprint from a glass or a hair from clothing

SOMETHING I KNOW

SOMETHING I KNOW???

- A password
  - A secret that is used for authentication
- A cryptographic secret
  - A secret that is combined with cryptographic algorithms
- Usually the weakest type of authentication
  - Secret might be copied without being detected
- But the most common type today

BREAKING PASSWORDS

BREAKING PASSWORDS

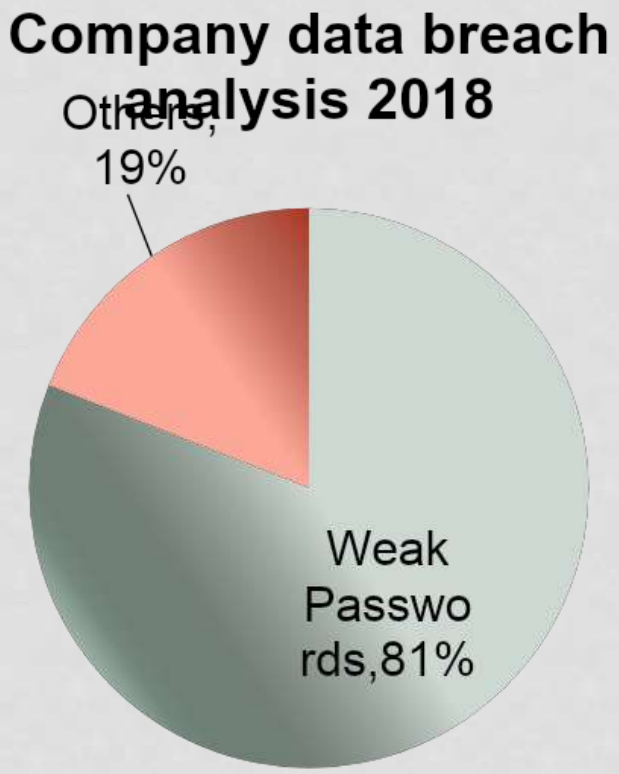
- The attacker uses knowledge about the user many users have passwords that are personal to them.
  - Name of husband, wife, daughter, son.
  - Name of family pet.
  - Name of the street or town where they live or where they were born.
  - Their house number, telephone number, car reg.

REALITY

- The attacker uses a dictionary a large percentage of passwords may be normal words.
- More chance that a user will have a password that is easy to remember than a random string.
- Attacker tries words from a dictionary.
- Could have millions of words

BREAKING PASSWORDS

- Top 25 passwords
  1. 123456 (Up 1)
  2. password (Down 1)
  3. 12345678 (Unchanged)
  4. qwerty (Up 1)
  5. abc123 (Down 1)
  6. 123456789 (New)
  7. 111111 (Up 2)
  8. 1234567 (Up 5)
  9. iloveyou (Up 2)
  10. adobe123 (New)
  11. 123123 (Up 5)
  12. Admin (New)
  13. 1234567890 (New)
  14. letmein (Down 7)
  15. photoshop (New)



PASSWORD ATTACKS

- Attack using brute force: try all combinations!
- Serious problem if users have short passwords
  - If we use only the letters a-z in lower case then:
  - Usually recommend 8 or more characters, mixed letters and numbers.
- Far slower than dictionary attack
- Longer keys are more difficult to attack

Letters a-z	**Combinations**
1	26
2	676
3	17576
4	456976
5	11881376
6	308915776

Key Size (bits)	Number of alternative keys	Average time required at one decryption per microsecond
40	$1.1 * 10^{12}$	6.36 days
56	$7.2 * 10^{16}$	1142 years
128	$3.4 * 10^{38}$	$5.4 * 10^{24}$ years
256	$1.1 * 10^{77}$	$1.7 * 10^{63}$ years

For comparison, the universe has been in existence less than 2x1010 years

CHOOSING GOOD PASSWORDS

CHOOSING GOOD PASSWORDS

- Never make it a real word - too vulnerable to dictionary attacks and guessing
- Always should be easy to remember and hard to guess
- Should be a combination of UPPER and lower case, numerals, letters and punctuation
- Should be reasonably long
  - 6 characters is only equivalent to a 40bit symmetric key
  - Need about 22 random characters to be equivalent to 128 bit key

SUGGESTIONS!!!

- Some suggestions
  - Use the first letters of a phrase e.g. use

    Tamflbitw from "The answer my friend is blowing in the wind"
  - Join several short words together e.g. iLoveNY!
  - Make up nonsense phrases e.g. Fip&Chips

PROTECTION : LOGIN PROCESS

- Make login process slow - e.g. add a 1 second delay between each login.
- Keep a record of failed login attempts and temporarily disable a login if a small limit is exceeded (ATM machines)
- Ask the user to carry out an action which requires human intervention
  - Recognising letters inside an image
  - Recognising a word in a sound file

PROTECTION : LOGIN PROCESS

SYMMETRIC AUTHENTICATION SCHEMES

- Most authentication schemes are based on a shared secret, where both parties know the same secret e.g.
  - One time passwords
  - Password-based authentication,
  - Symmetric encryption based schemes: Kerberos

ONE TIME PASSWORD - SOFTWARE

- "One time password" schemes use a shared secret (e.g. a password) and an encryption algorithm to produce an encrypted secret that is used just once as a one time password
- S/Key from Bellcore - Internet RFC 1760
- User's password and seed are hashed multiple times using MD4 hashing algorithm

SOMETHING I KNOW

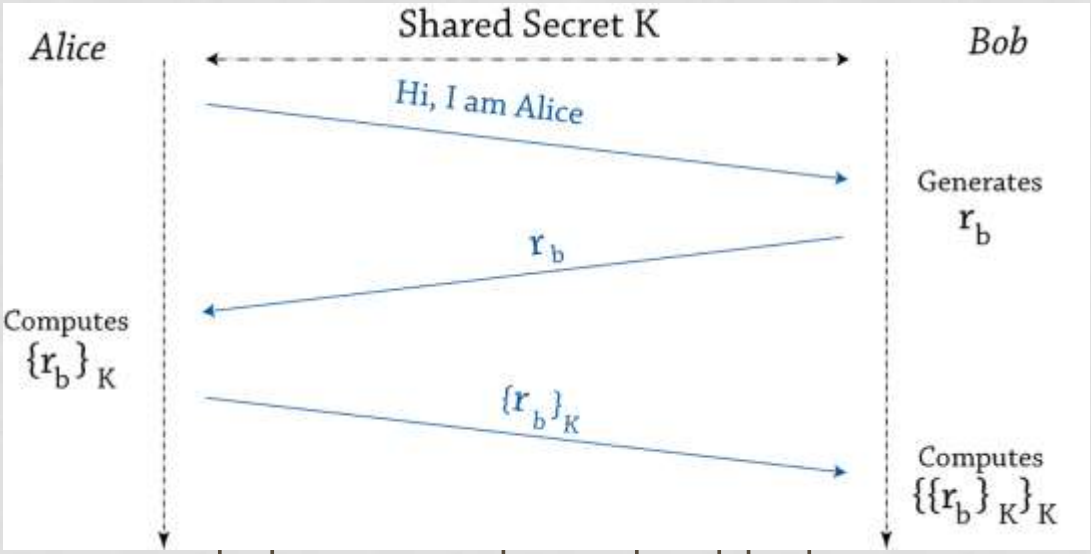
PASSWORD BASED AUTHENTICATION

PASSWORD BASED AUTHENTICATION

- Based on usernames and passwords
- Password may be passed
  - As a clear text
  - Or hashed (along with a "salt" word)
  - Or encrypted (and salted)
- Often open to sniffing attacks, dictionary attacks and pre-computed hash attacks
- No Internet standards can now be based on clear passwords

PASSWORD BASED AUTHENTICATION

- Symmetric Cryptography
  - Strong authentication: Alice proves to Bob that she know a pre shared secret without revealing it (challenge/response)



  - If Alice wants to prove her identity to n users, she stores n shared secret
    - Q: Pick a new random each time, why?

DISTRIBUTED SYSTEMS AND PASSWORD AUTHENTICATION

- How can I gain access to multiple computer systems if password based authentication is used? Possible solutions:
  - Multiple passwords, one for each system
  - Use same password in all systems?
  - Single sign-on application that stores the passwords for each system and has one for itself
    - E.g. Mozilla firefox
  - Single sign-on where password is stored in just one system and other systems trust this one to perform the authentication properly on their behalf (e.g. Microsoft Passport, Shibboleth)

THE MUTUAL AUTHENTICATION PROBLEM

- How can two people authenticate to each other using passwords?
- If a set of N persons mutually share symmetric secrets, the system will need N(N-1)/2 keys
  - Scalability issues

SOMETHING I KNOW

KERBEROS

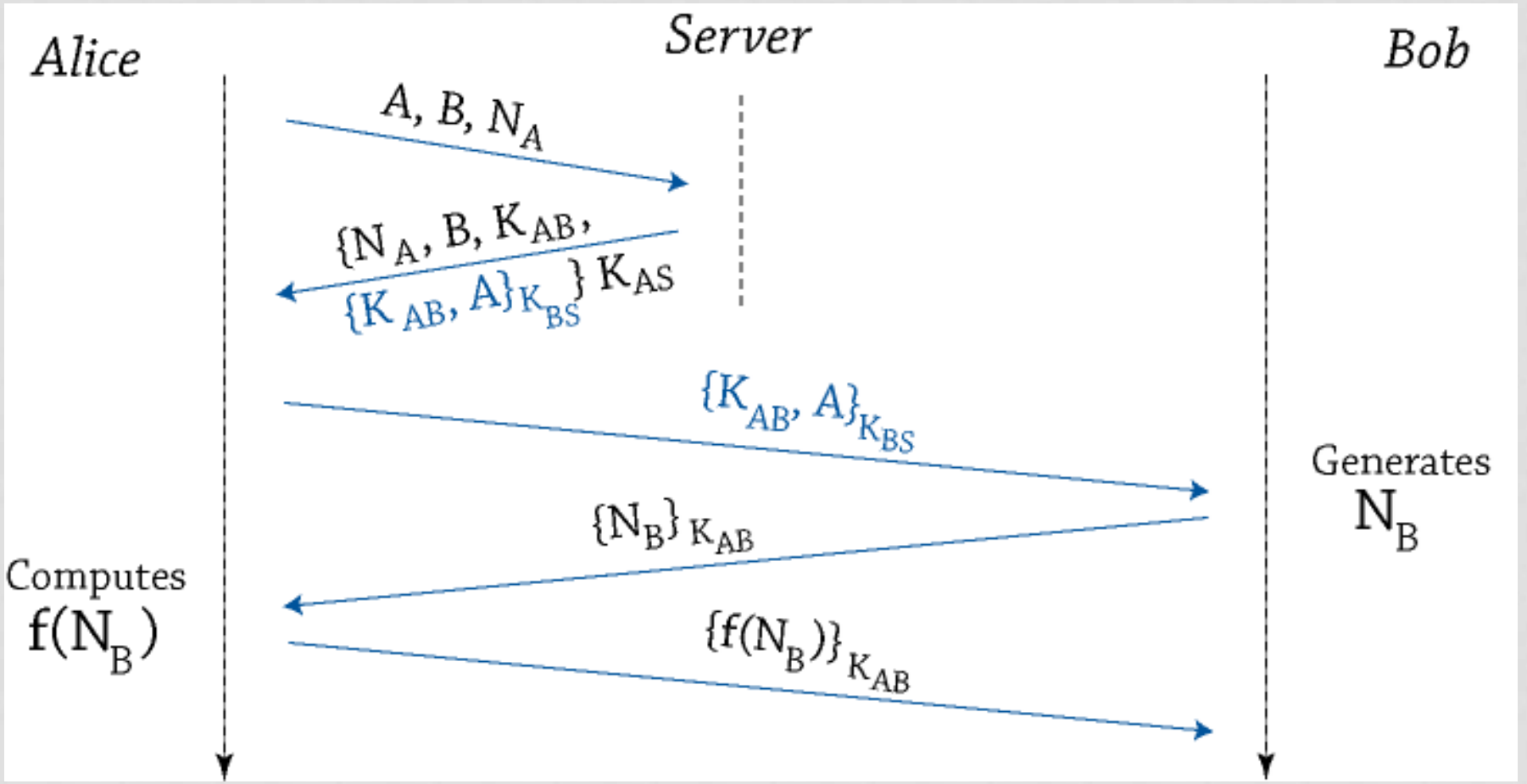
NEEDHAM-SCHROEDER

- Is based on encrypting a message with a shared secret, so that only the other party knowing the secret can decode the message.
  - But cannot share secrets with everyone – multiple password problem again. Therefore share with just one party – a secret key server.
- Trusted symmetric key server shares a secret key with every user.
- When Alice wants to talk to Ben she asks the key server for a new secret key.
- The key server generates a new key, encrypts it with Alice's secret and sends it to Alice. It also includes the new key encrypted with Ben's secret.
- Alice decrypts the message and forwards Ben's encrypted key to him
- Ben decrypts the new key and sends a confirmation message to Alice encrypted with the new key
- Alice decrypts this, performs a simple operation on the message and sends it encrypted back to Ben
- Both sides now know that they share the same secret

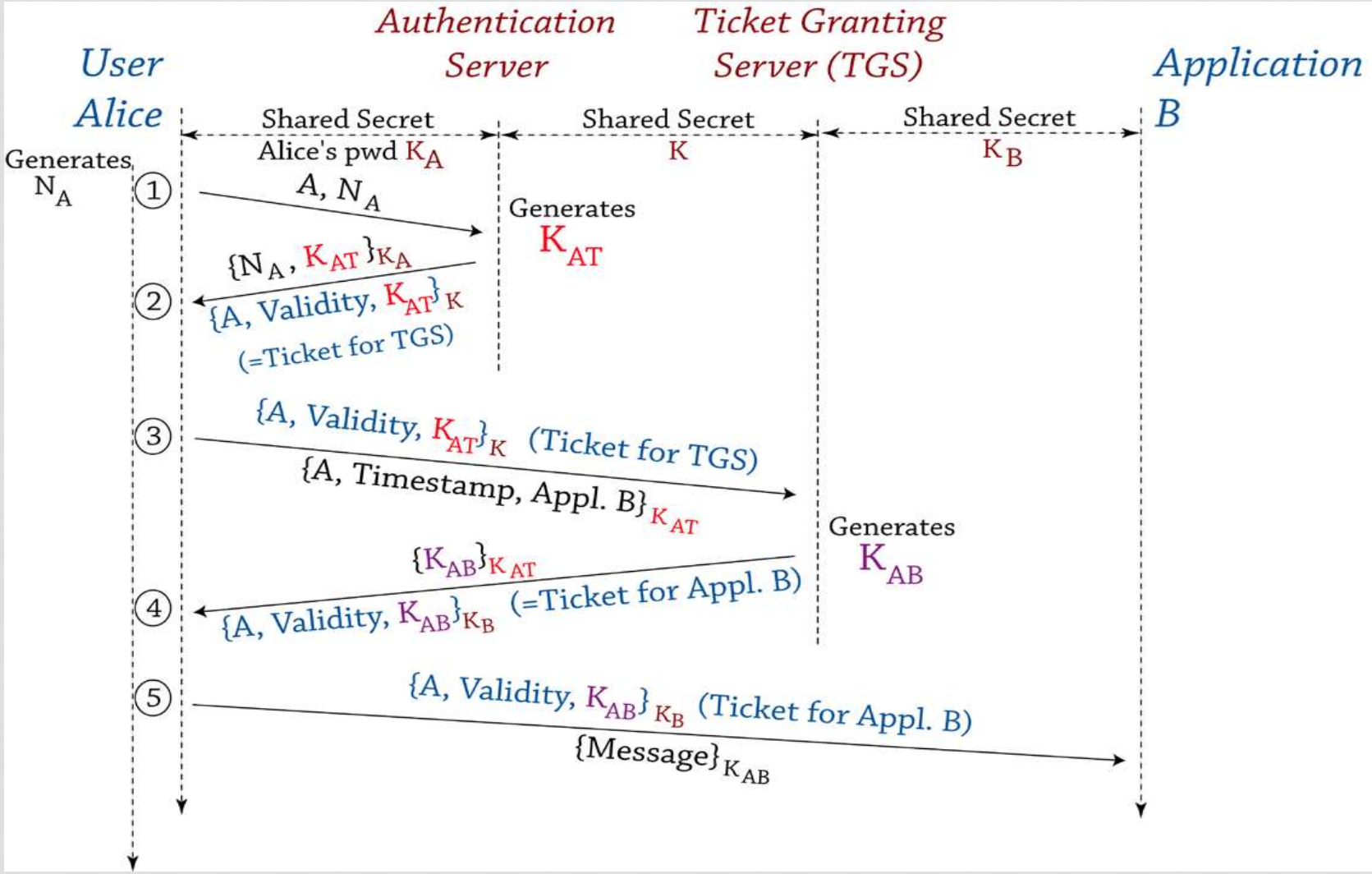
NEEDHAM-SCHROEDER

- 1. Alice->Server: A, B, Na.
- 2. Server->Alice: {Na, B, Kab, {Kab, A}Kbs}Kas
- 3. Alice->Ben: {Kab,A}Kbs
- 4. Ben->Alice: {Nb}Kab
- 5. Alice->Ben: {f(Nb)}Kab
  - Where A=name of Alice
  - B=name of Ben
  - Na=nonce generated by Alice
  - Nb=nonce generated by Ben
  - Kab=New secret key to be used by Alice and Ben
  - Kbs=Ben's secret shared with the server
  - Kas= Alice's secret shared with the server
  - F(Nb)= a simple function applied to Nb such as subtracting 1

NEEDHAM-SCHROEDER



KERBEROS


User
Alice
Authentication
Server
Ticket Granting
Server (TGS)
Application
B
Shared Secret
Alice's pwd K_A
Shared Secret
K
Shared Secret
K_B
Generates
N_A
1
A, N_A
Generates
K_{AT}
$\{N_A, K_{AT}\}_{K_A}$
2
$\{A, Validity, K_{AT}\}_K$
(=Ticket for TGS)
3
$\{A, Validity, K_{AT}\}_K$ (Ticket for TGS)
$\{A, Timestamp, Appl. B\}_{K_{AT}}$
Generates
K_{AB}
$\{K_{AB}\}_{K_{AT}}$
4
$\{A, Validity, K_{AB}\}_{K_B}$ (=Ticket for Appl. B)
5
$\{A, Validity, K_{AB}\}_{K_B}$ (Ticket for Appl. B)
$\{Message\}_{K_{AB}}$

SUMMARY OF PROBLEMS WITH SHARED SECRET/SYMMETRIC ENCRYPTION TECHNIQUES

- They need a shared secret, and each pair of users needs a different shared secret
- Secret key management – how to reliably share the secret without anyone else eavesdropping
- Not easily scalable
- Possible solutions:
  - Use a key server such as in N-S and Kerberos
  - Use a Key exchange protocol such as Diffie Hellman
  - Use asymmetric encryption

DIFFIE HELLMAN

DIFFIE HELLMAN KEY EXCHANGE PROTOCOL

- Allows two people to calculate a shared secret, without eavesdroppers being able to work out the shared secret.
- It works using complicated mathematics based on an exponential equation and modulus $X=Y^A$ Mod P
- A group of people agree on values for Y and P
- If two people want to share a secret, they each choose different values for A (A_1 and A_2) and send the result of the calculation, X_1 and X_2 to each other.

DIFFIE HELLMAN KEY EXCHANGE PROTOCOL

- They then perform the following calculation $S=X_i^A \text{ Mod } P$ using their own A and the received X_i , and each derives the same secret S, which no-one else can do because no one else knows either A_1 or A_2

- Proof. $S=X_1^{A_2} \text{ Mod } P = X_2^{A_1} \text{ Mod } P = Y^{A_1A_2} \text{ Mod } P$

END