Unit-VI

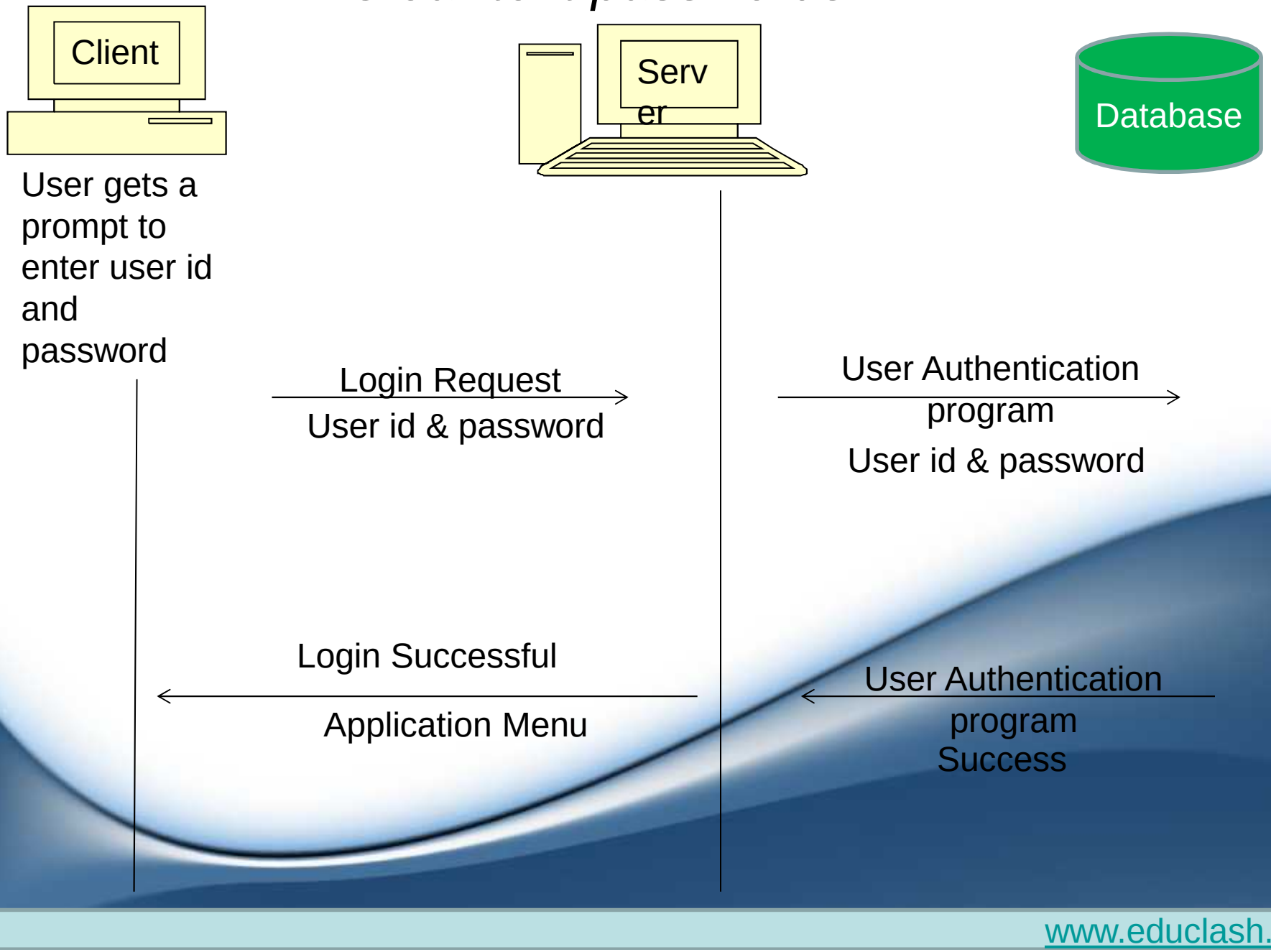
User Authentication Mechanisms

- Authentication is the first step in any cryptographic solution
- ***Authentication can be defined as determining an identity to the required level of assurance***

Passwords

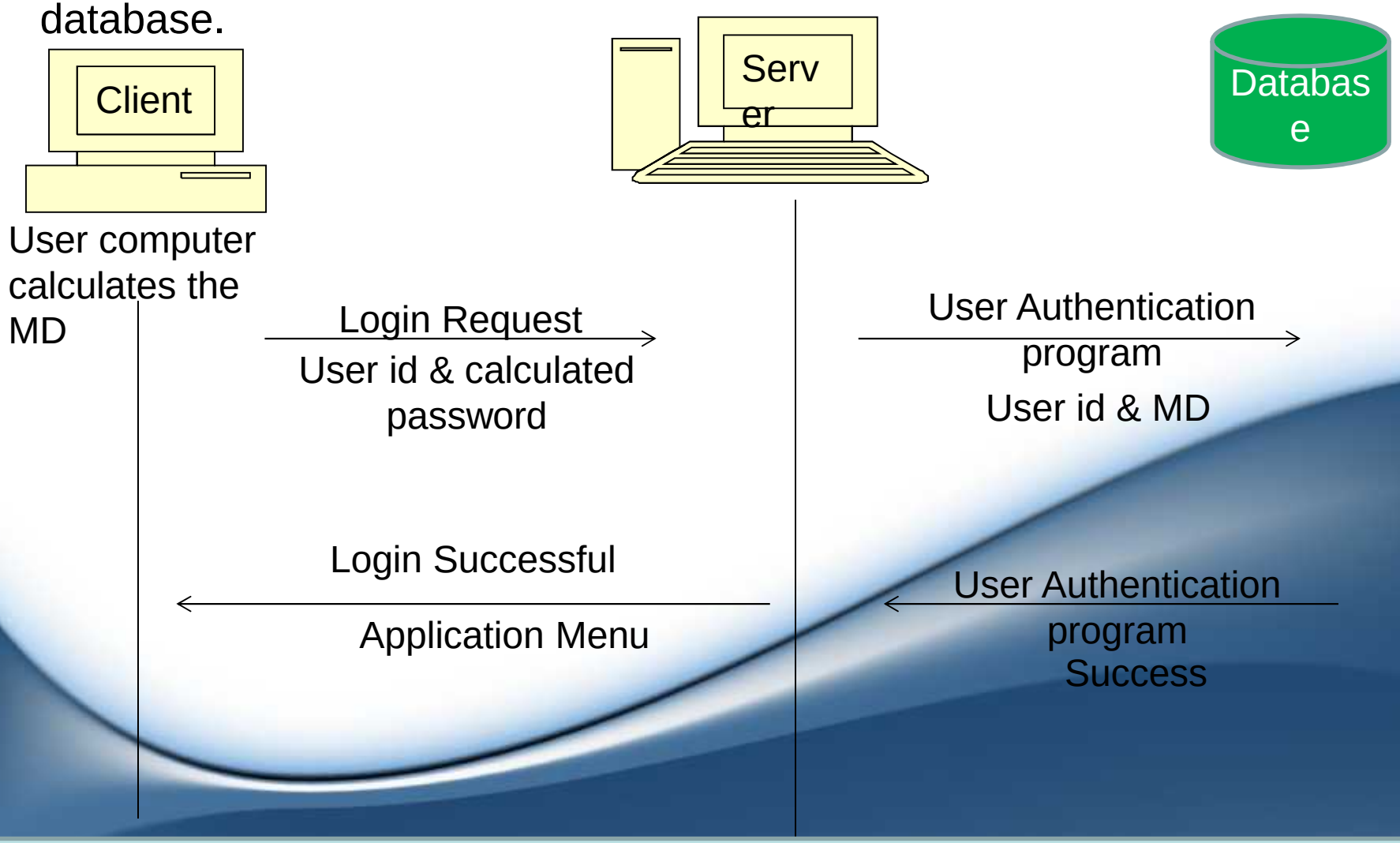
- *A password is a string of alphabets, numbers and special characters, which is supposed to be known only to the entity (usually a person) that is being authenticated.*

Clear text passwords:

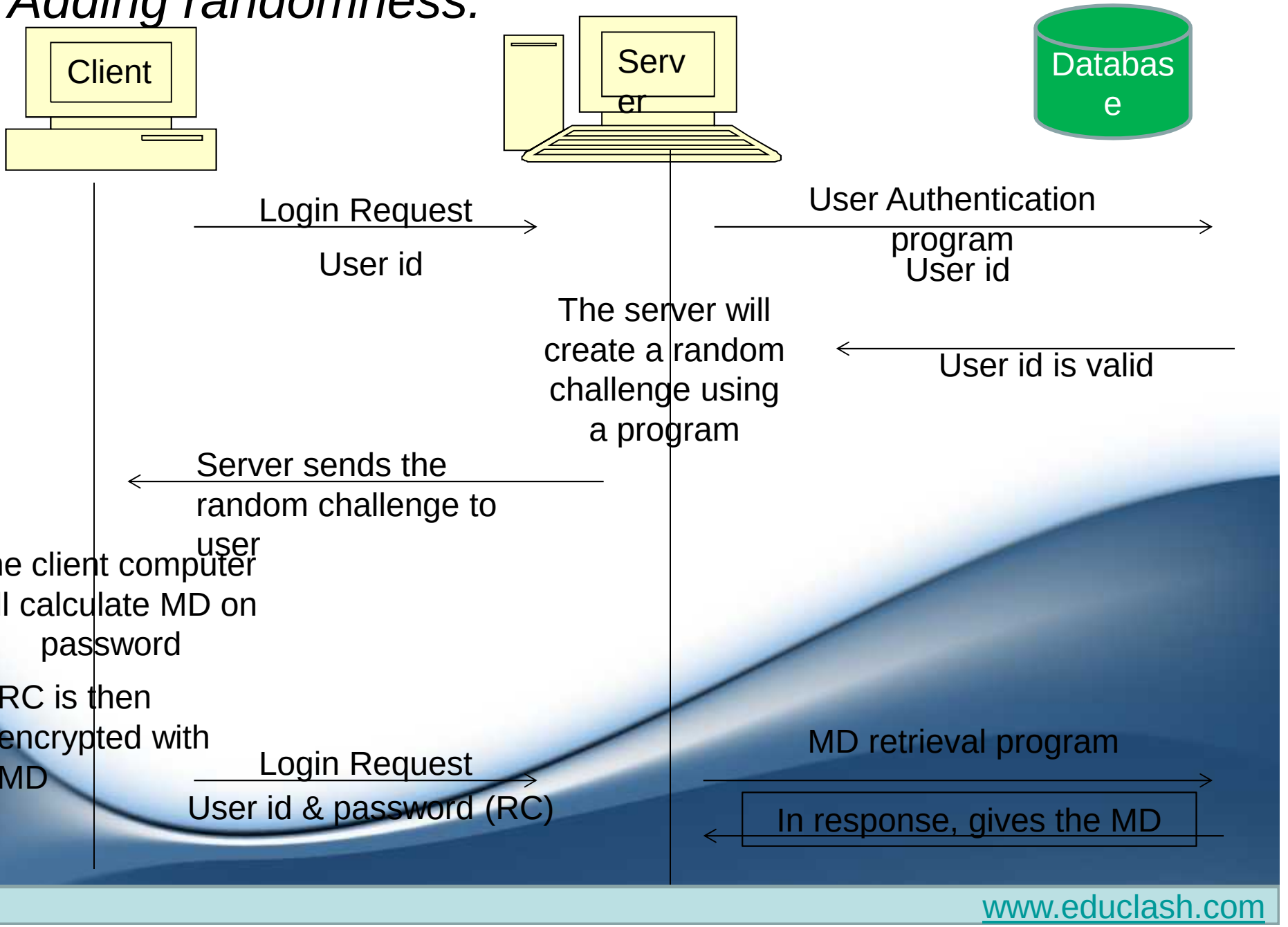


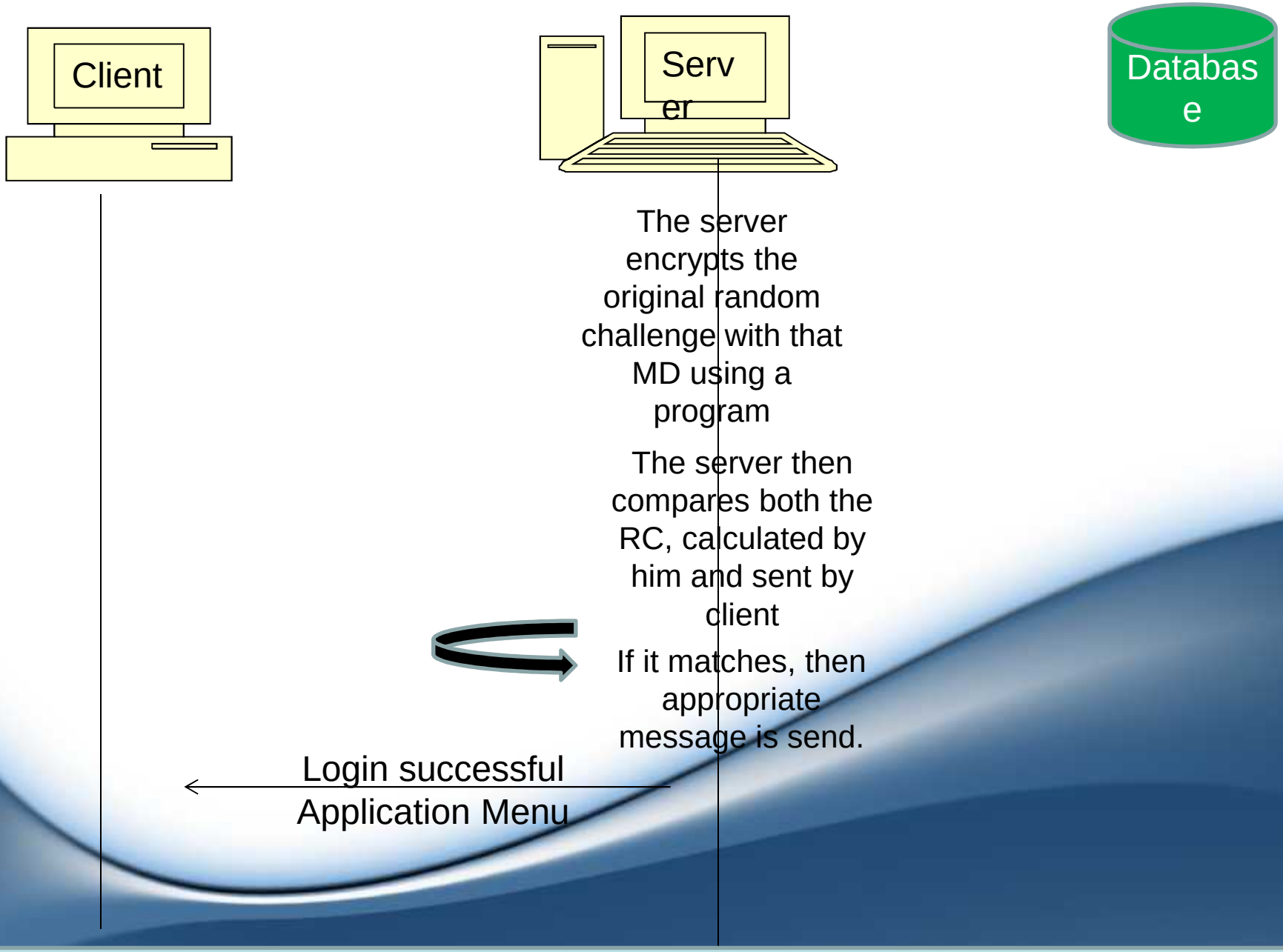
Something derived from passwords:

Step:1 Create a message digest of passwords
Step: 2 Store the user id and message digest of the passwords in the user database.



Adding randomness:




Client
Serv
er
Databas
e
The server encrypts the original random challenge with that MD using a program
The server then compares both the RC, calculated by him and sent by client
If it matches, then appropriate message is send.
Login successful
Application Menu

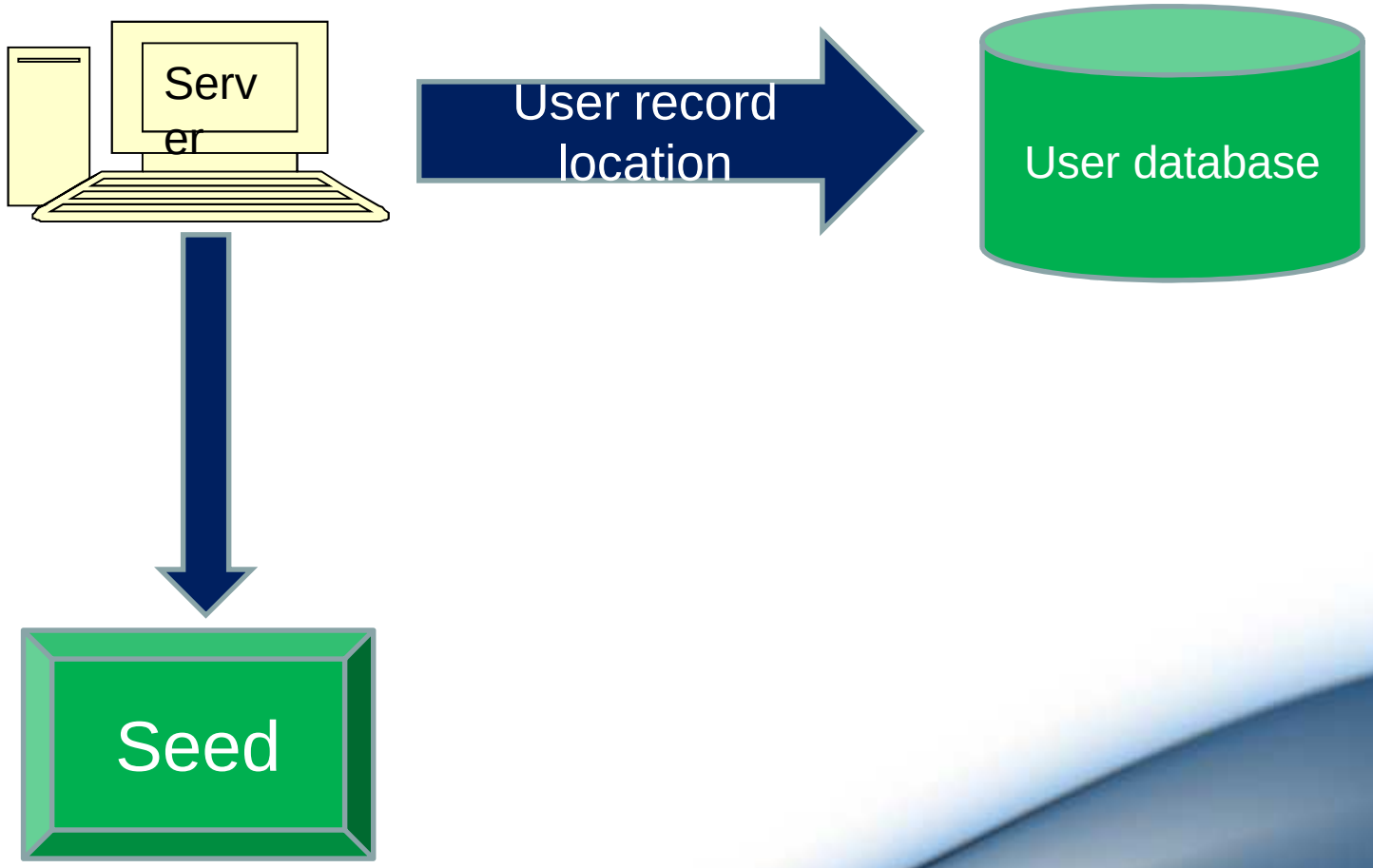
Authentication Tokens

- An authentication token is a small device that generates a new random value every time it is used.
- The small devices are typically of the size of small key chains, calculators or credit cards.
- Usually, an authentication token has the following features:-
- Processor
- Liquid Crystal Display(LCD)
- Battery
- (Optional) a small keyboard for entering information
- (optional) a real time clock.
- Each authentication token is pre-programmed with a unique number called as random seed (or seed)

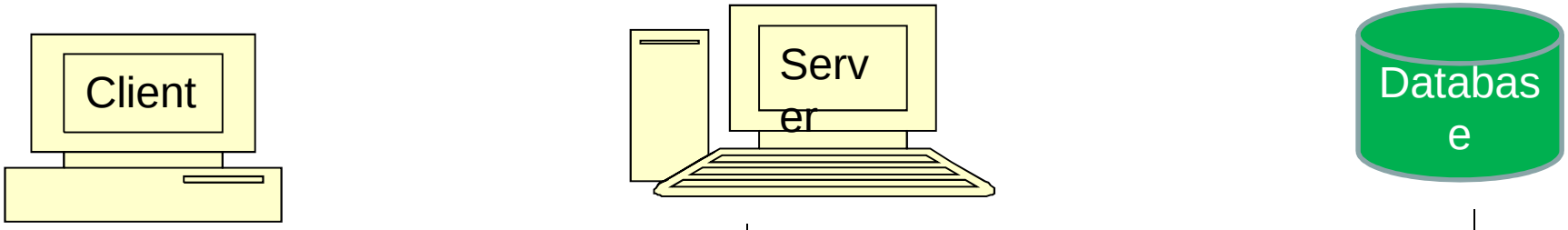

RSA
SecurID
684 314.
Secured by RSA


RSA SecurID
246073.
1 2 3 4 5 6 7 8 9 0


RSA



Step: 1 Whenever an authentication is created , the corresponding random seed is generated for the token by the Authentication Server. This seed is stored or pre-programmed inside the token and also stored in database.



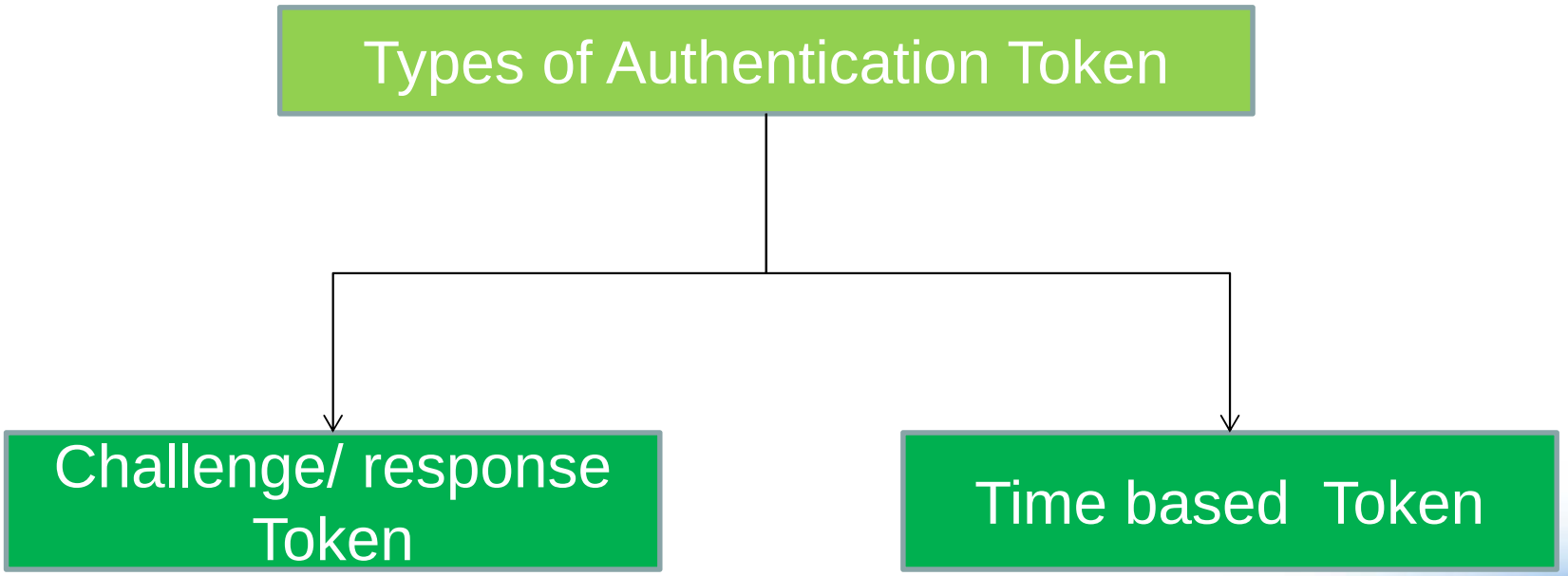
User id and one-time password obtained by token are sent to server

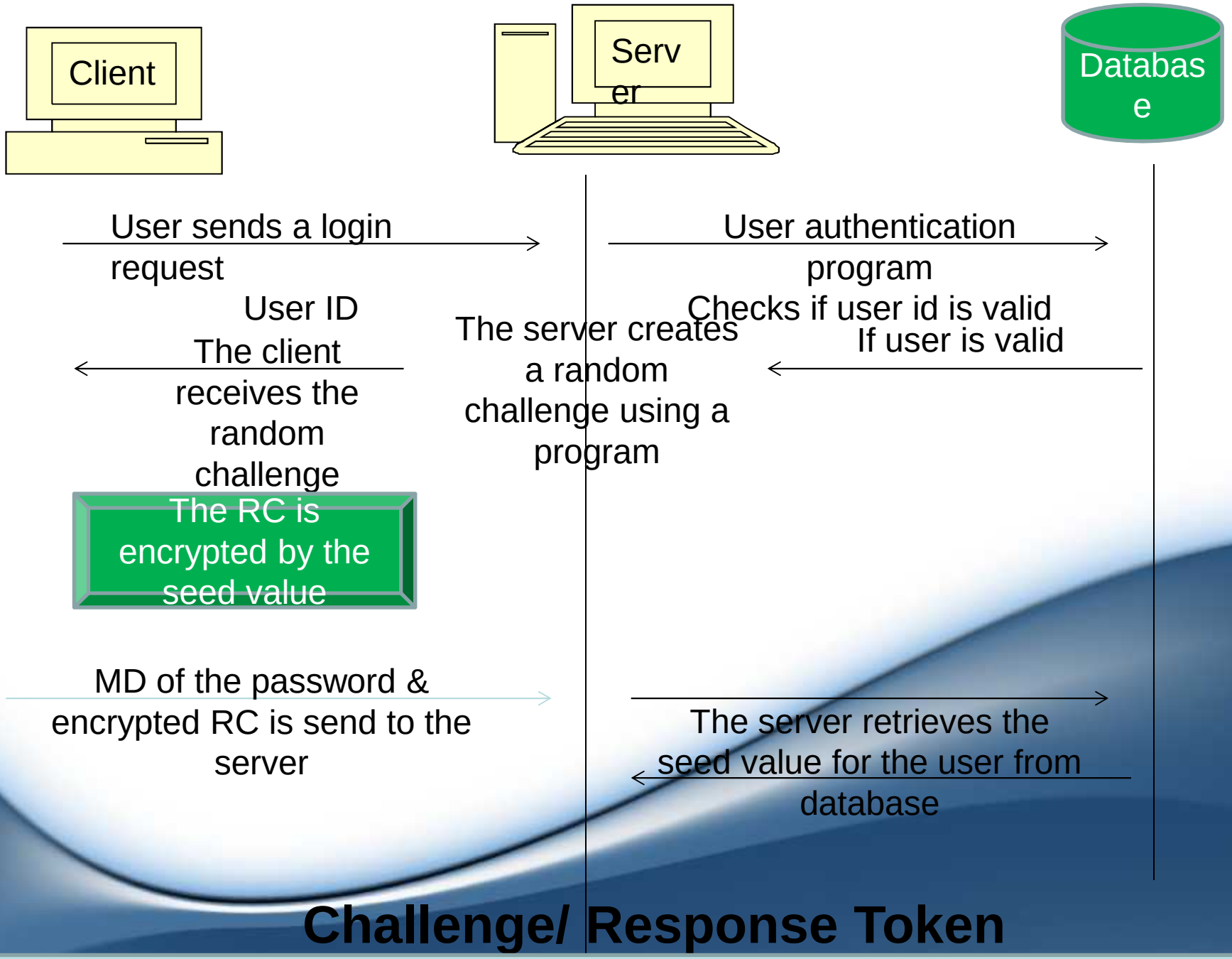
The server's seed retrieval program now retrieves the seed from the database

Database sends the seed related to user id

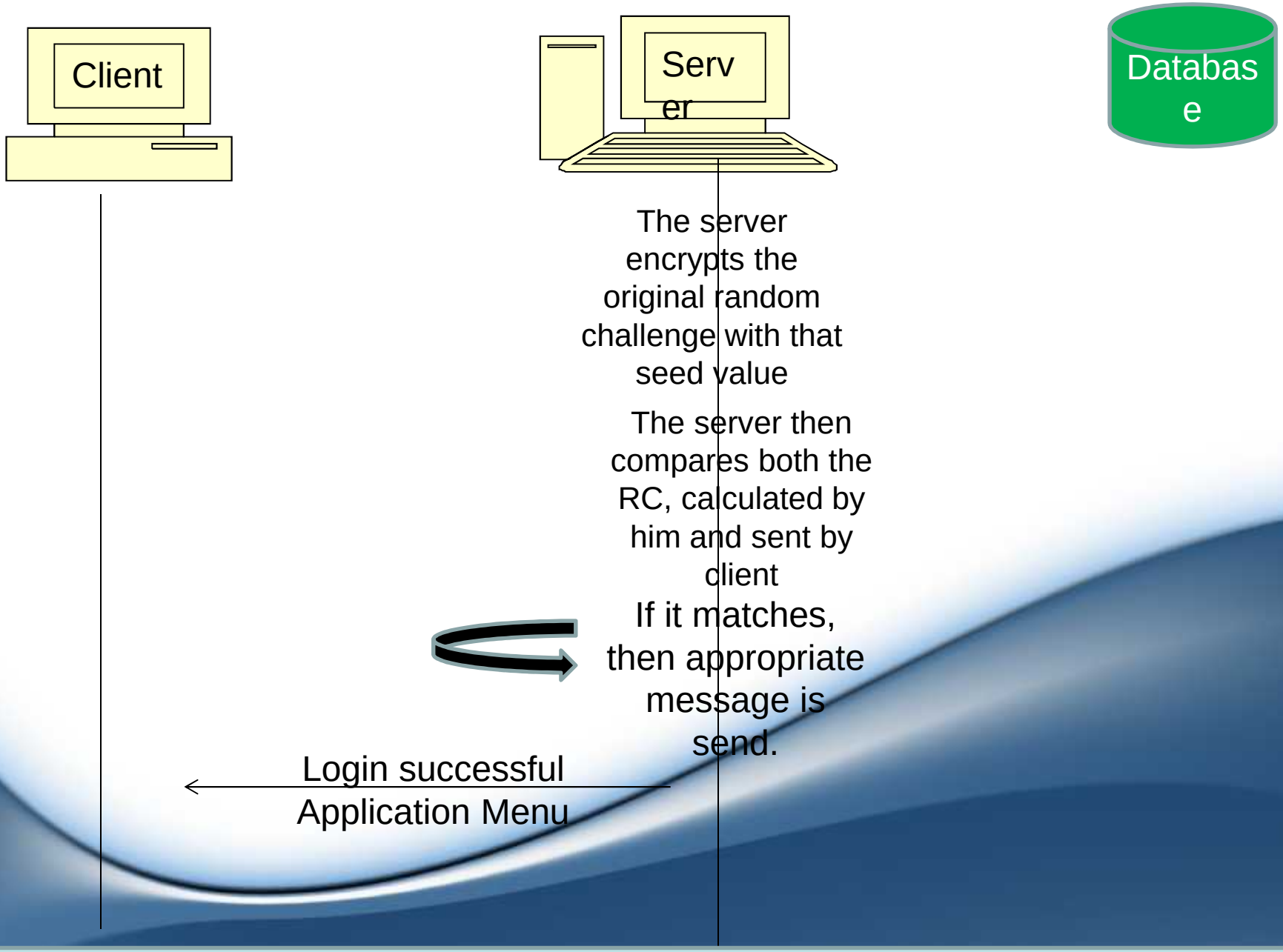
The server password validation program calculates the one-time password and checks the seed against the password

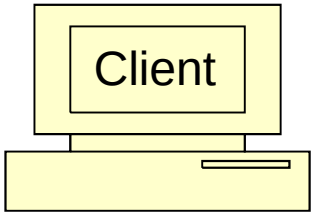
If the password is correct, then appropriate message is send to user

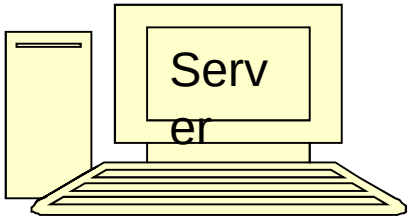

Types of Authentication Token
Challenge/ response Token
Time based Token



Challenge/ Response Token


Client
Serv
er
Databas
e
The server encrypts the original random challenge with that seed value
The server then compares both the RC, calculated by him and sent by client
If it matches, then appropriate message is send.
Login successful
Application Menu





The seed value and the system time of token, together perform cryptographic algorithm to generate a password automatically.

User sends a login request

User ID & password

The server retrieves the seed value for the user from database

The server calculates the same cryptographic function as was done by the token using seed value & system time

The server compares both the passwords

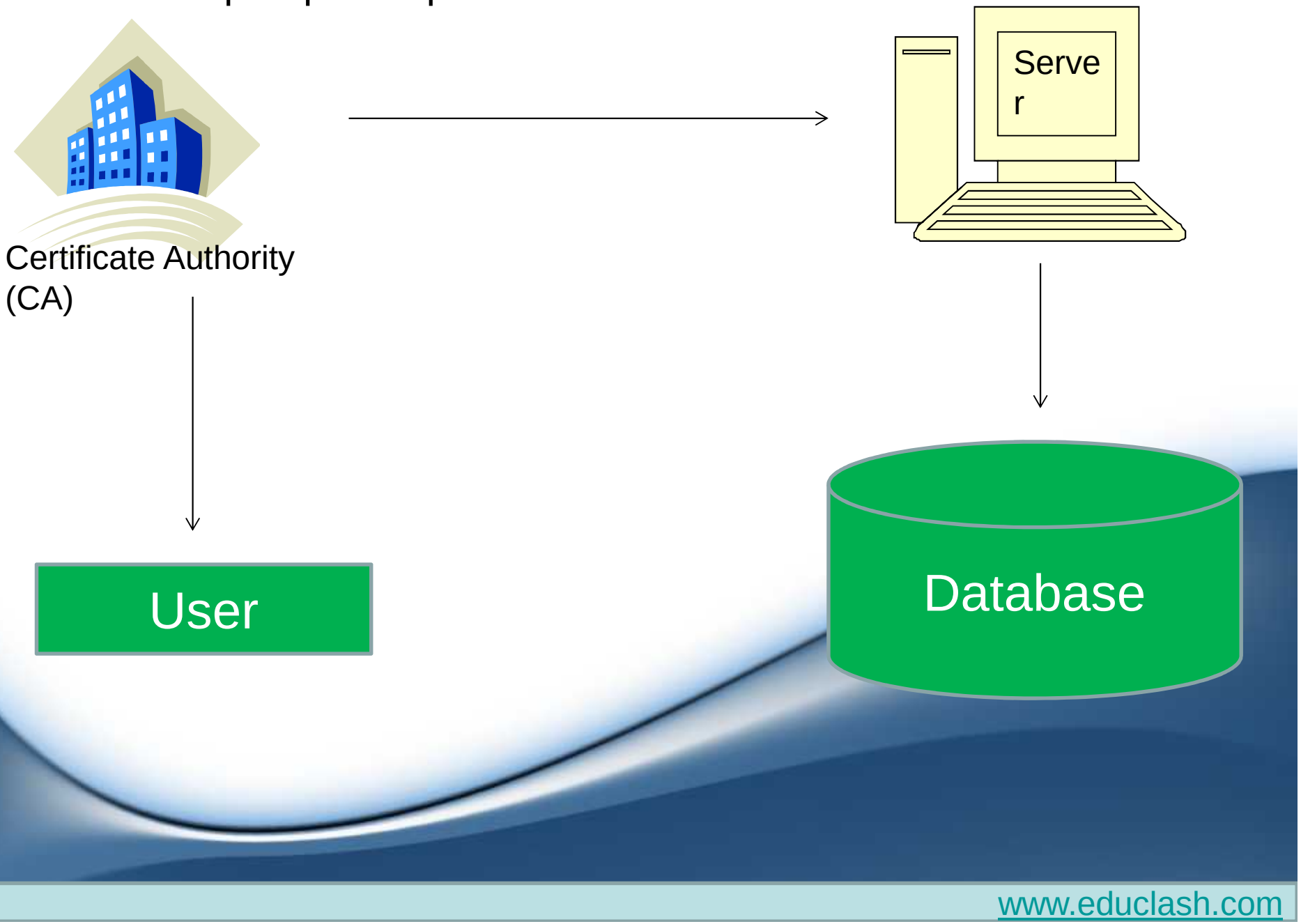
If it matches, then appropriate message is send.

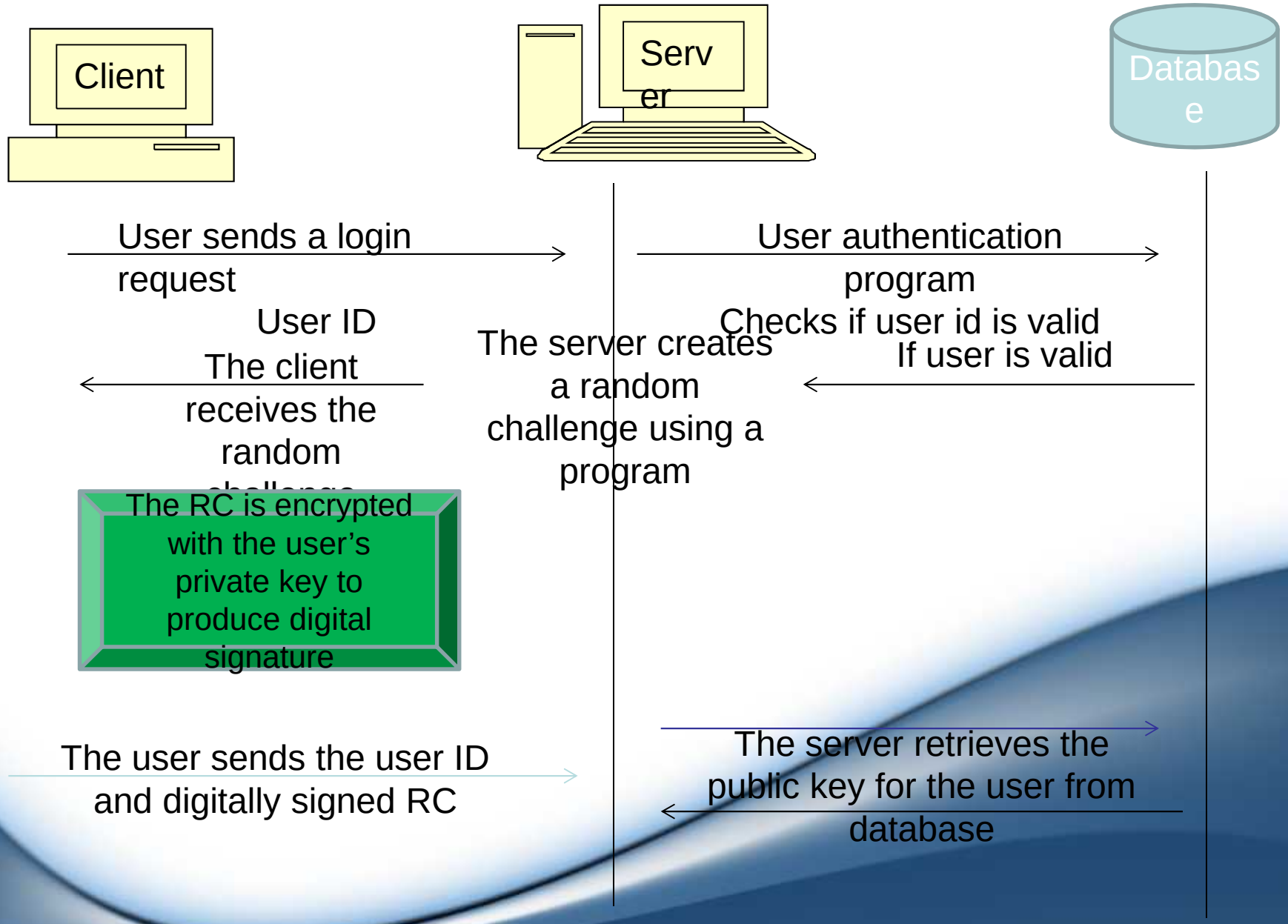
Time Based Token

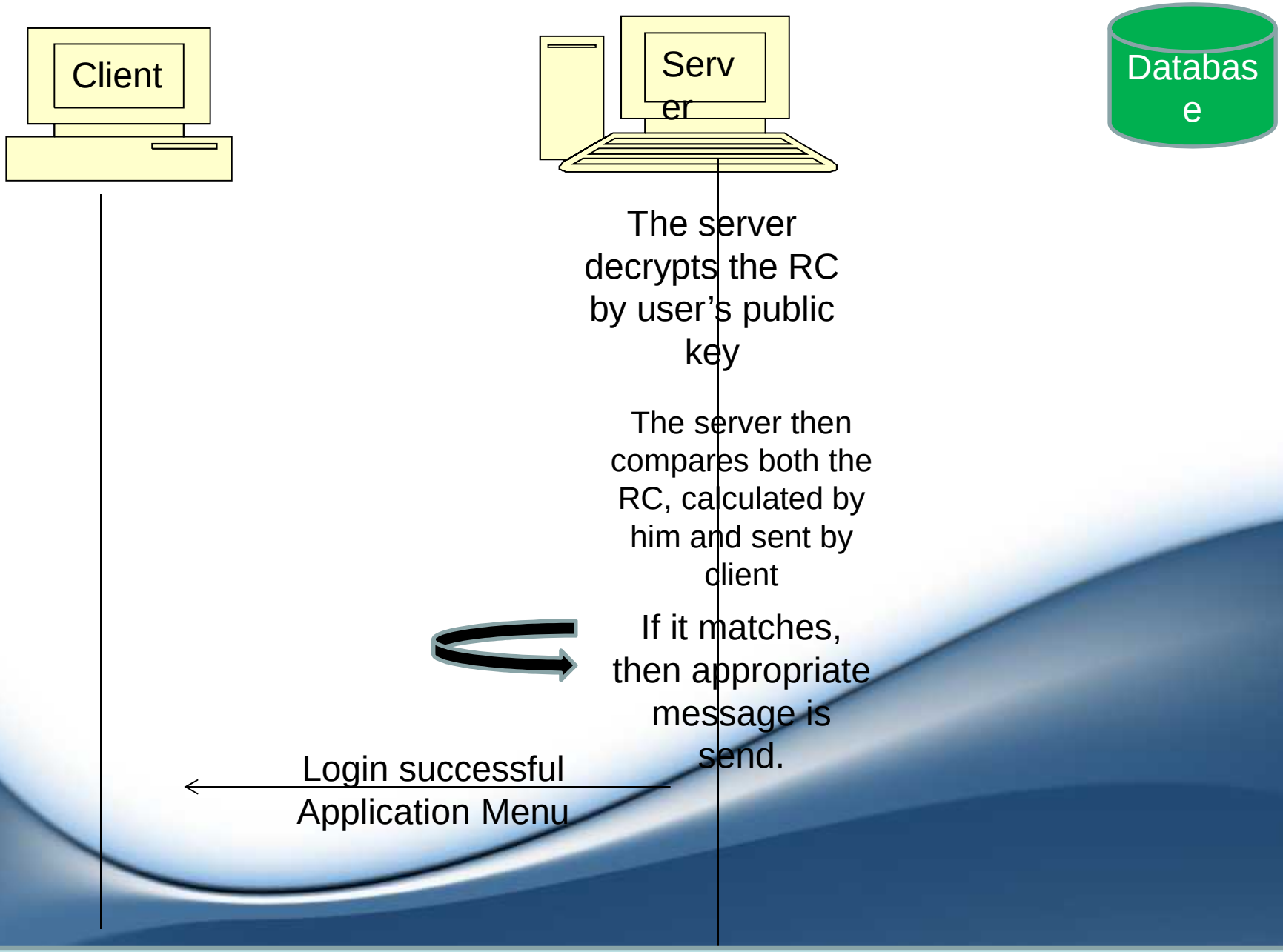
Certificate Based Authentication

- This is based on digital certificate of a user.
- Here the user is expected to *have* something and *know* something.

Step: 1 pre-requisite in Certificate based authentication




Client
Serv er
Databas e
User sends a login request
User authentication program
Checks if user id is valid
If user is valid
User ID
The client receives the random
The server creates a random challenge using a program
The RC is encrypted with the user's private key to produce digital signature
The user sends the user ID and digitally signed RC
The server retrieves the public key for the user from database


Client
Serv
er
Databas
e
The server decrypts the RC by user’s public key
The server then compares both the RC, calculated by him and sent by client
If it matches, then appropriate message is send.
Login successful
Application Menu

Biometric Authentication

- A biometric device works on the basis of human characteristics.
- Every time, the sample produced in biometric varies slightly.
- That is why many biometric samples are combined and their average is stored in the database.
- Thus an approximate match is acceptable.
- In biometric authentication system, 2 configurable parameters are defined:
- FAR (False Accept Ratio) = is a measurement of the chance that a user who should be rejected is actually accepted by the system as *good enough*.
- FRR (False Reject Ratio) = is a measurement of the chance that a user who should be accepted as valid is actually rejected by the system as *not good enough*.
- If the two samples match to the expected degree on the basis of values of FAR and FRR, the user is authenticated otherwise not.

Biometric Techniques

Physiological techniques

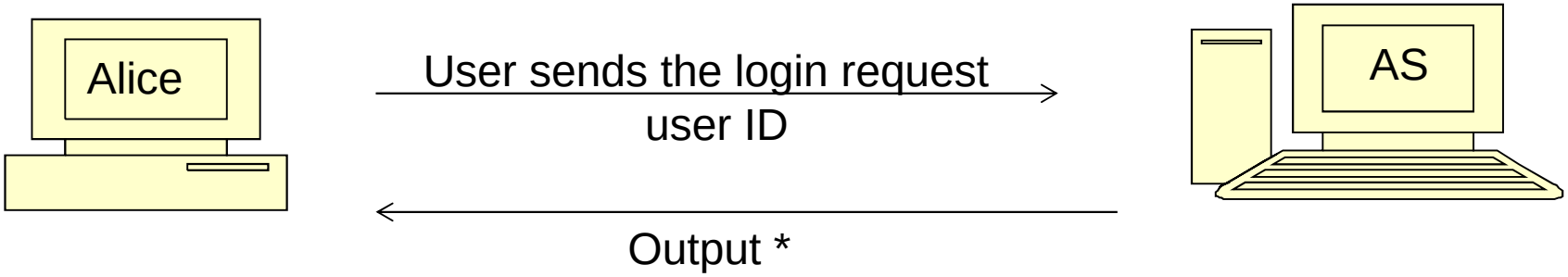
1. Face – check and measure the distance between the various facial features such as eyes, nose & mouth.
2. Voice – characteristics of sound waves, pitch and tone of voice.
3. Fingerprint – authentication uses 2 approaches:
Minutiae based: graph of individual ridge positions is drawn
Image based: image of fingerprint is stored.
4. Iris – unique pattern of inside the iris
5. Retina – the vessels carrying blood supply at the back of human eye are examined

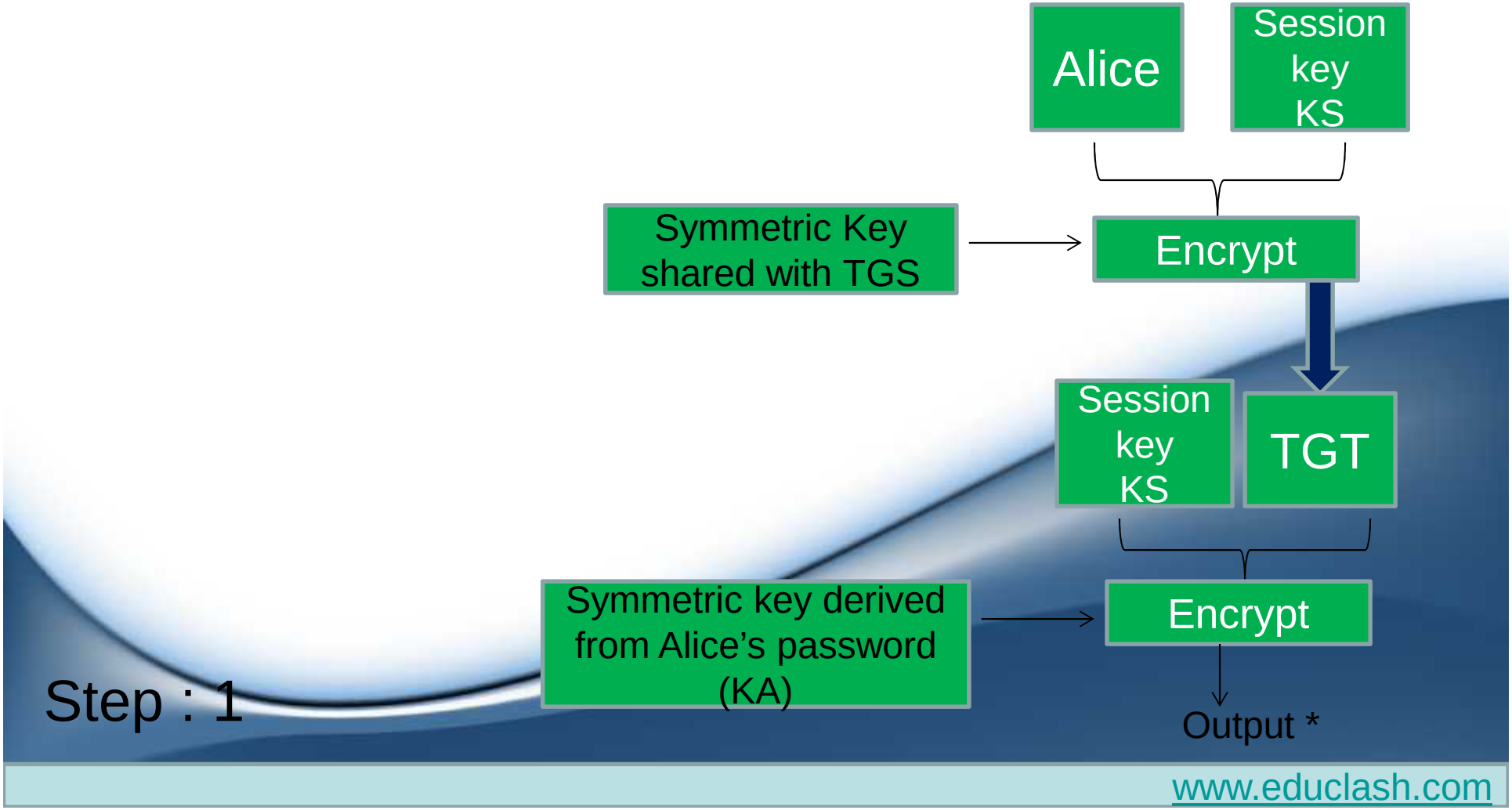
Behavioral techniques

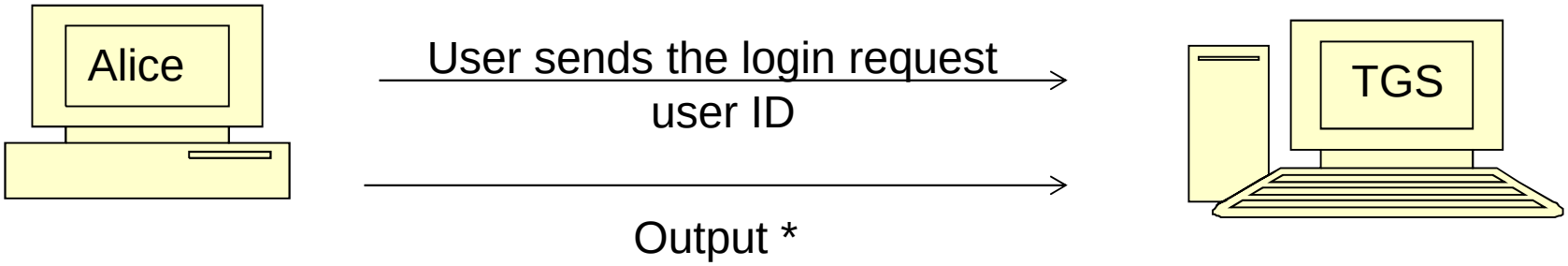
1. Keystroke – speed of typing, strength of keystrokes, time between 2 keystrokes, error percentage and frequency.
2. Signature – physically signed by the authorizer is compared by the computerized scanned copy

Kerberos

- Kerberos is based on an algorithm called Needham-Shroeder.
- Designed at MIT.
- Kerberos signifies a multi headed dog in Greek mythology.
- There are 4 parties involved in Kerberos:
- Alice: the client workstation
- Authentication Server (AS) :verifies(authenticates) the user during login.
- Ticket Granting Server (TGS): issues tickets to certify *proof of identity*.
- Bob: the server offering services such as network printing, file sharing or an application program.


Alice
User sends the login request
user ID
AS
Output *


Alice
Session key KS
Symmetric Key shared with TGS
Encrypt
Session key KS
TGT
Symmetric key derived from Alice's password (KA)
Encrypt
Output *
Step : 1



Encrypt

Session key (KS)

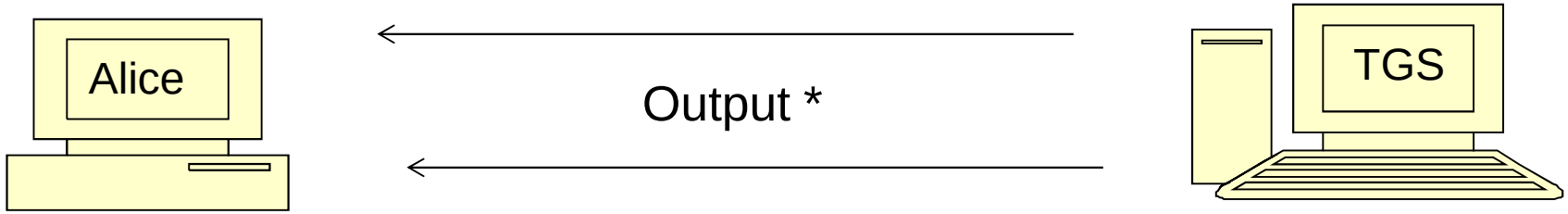
Encrypted Time

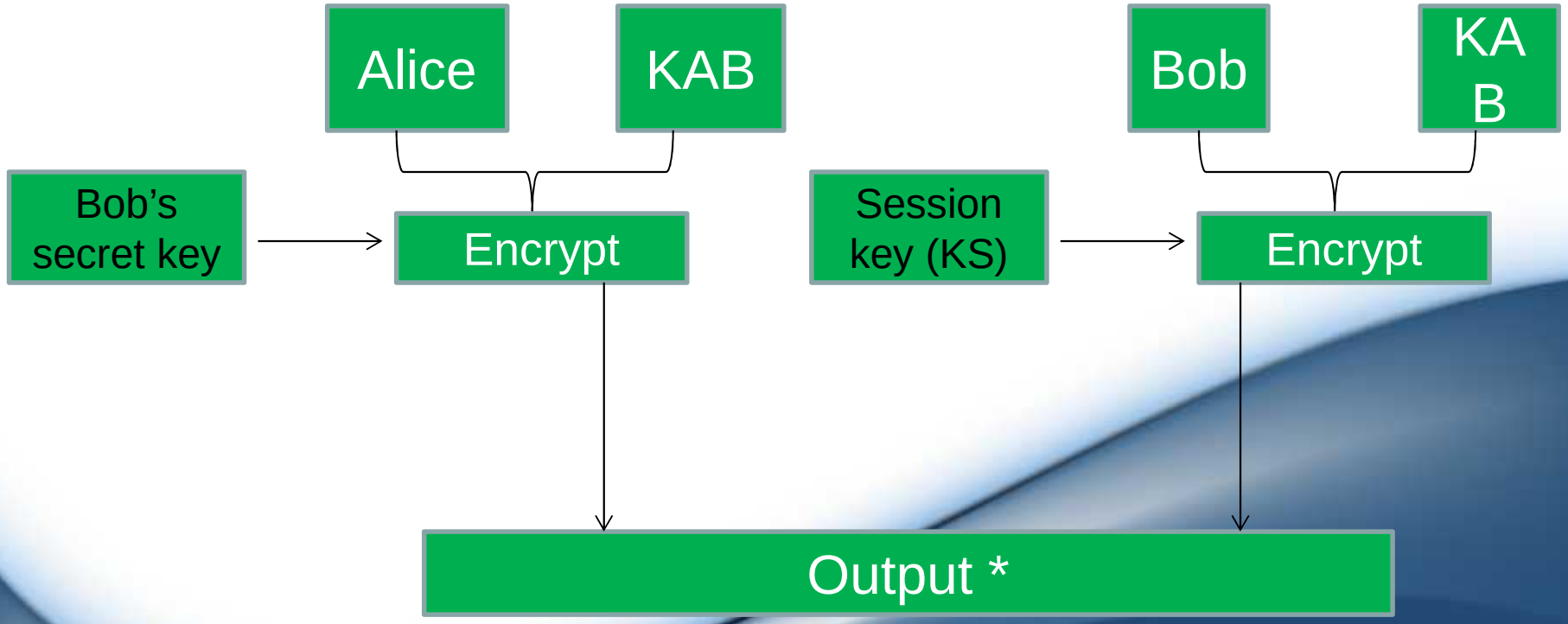
TGT

Bob Id

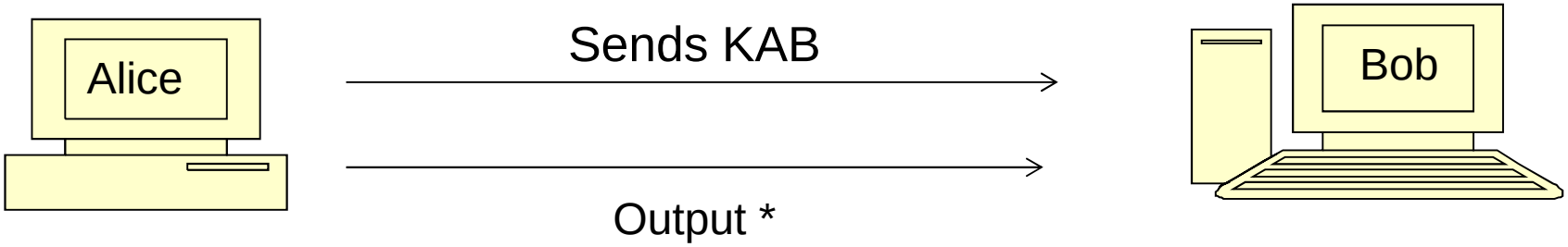
Output *

Step : 2 Alice wants to communicate with Bob(email server)





In response the TGS will send response back to Alice



Encrypt

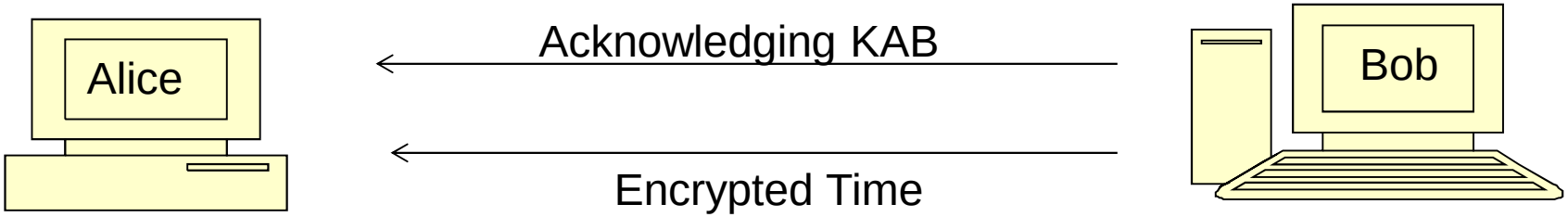
KAB

Encrypted Time

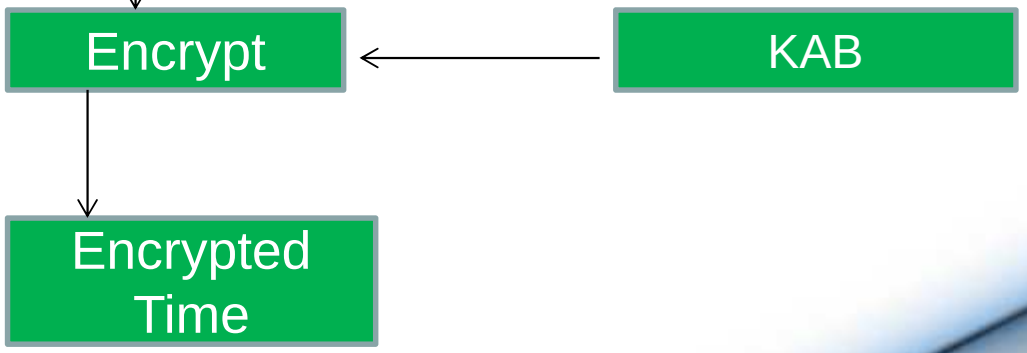
(Alice +KAB) encrypted with Bob's secret key

Output *

Step : 3 Alice contacts Bob for accessing the server



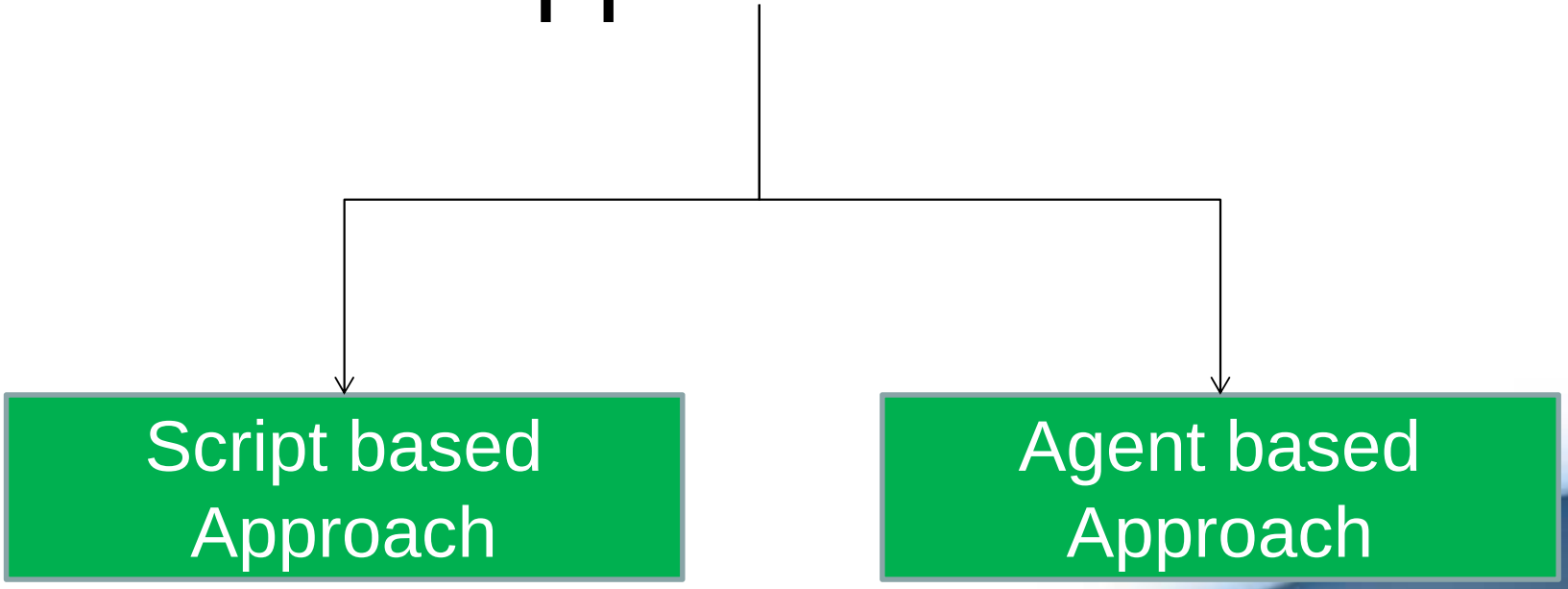
Bob adds 1 to the timestamp sent by Alice



Bob acknowledges the receipt of KAB

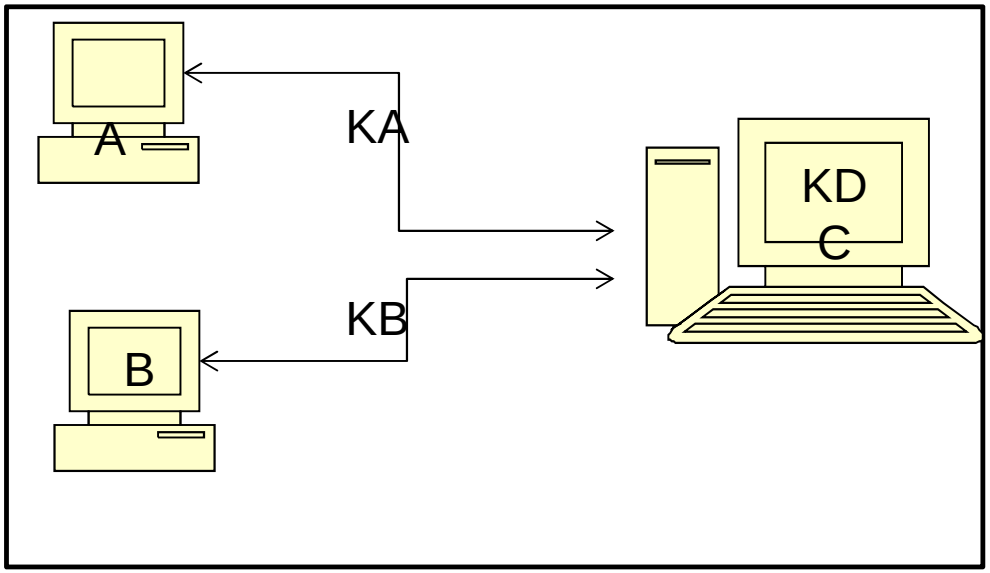
- If Alice wants to continue communicating with Bob alone, she need not obtain a new ticket every time.
- Since Alice needs to authentic only once, this mechanism is called Single Sign On(SSO).
- If Alice now wants to communicate with another server say Carol:
- Alice needs to obtain another shared key from TGS.

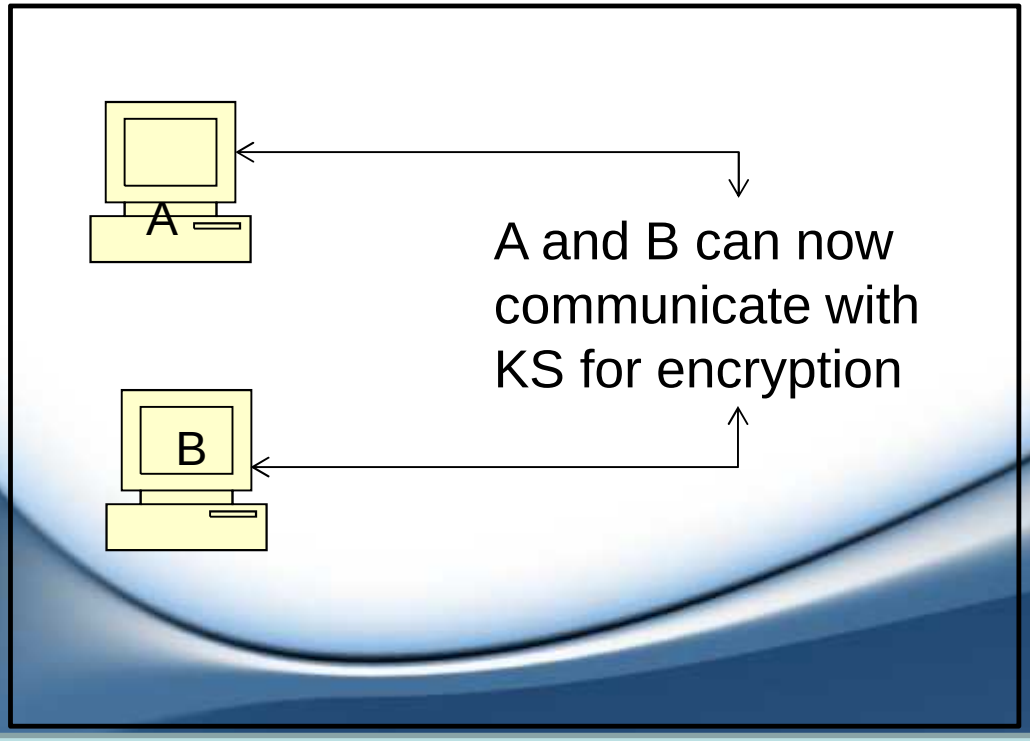
Single Sign On (SSO) Approaches

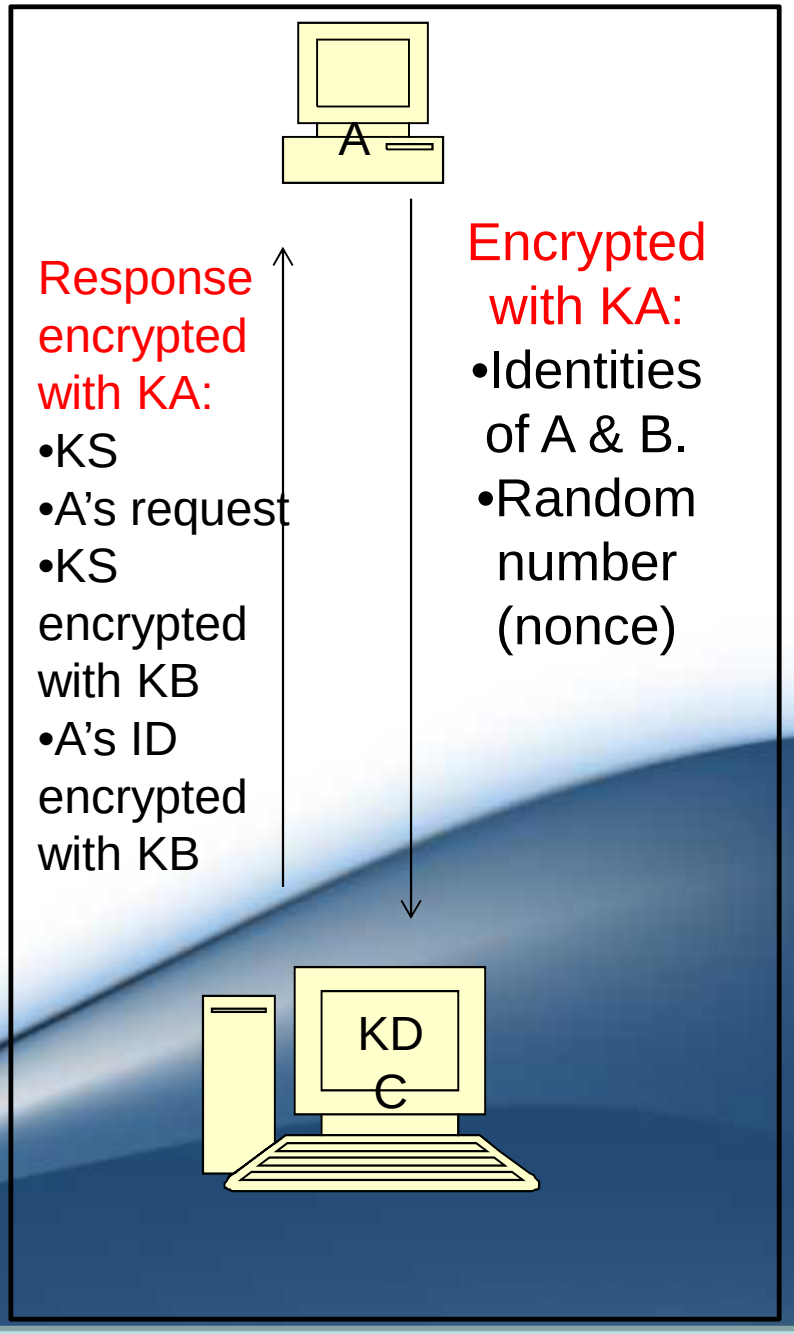


Key Distribution Center (KDC)

- KDC is a central authority dealing with keys for individual nodes in a computer network.
- It is similar to the concept of AS and TGS in Kerberos.
- The basic idea is that every node shares a unique secret key with KDC.
- Whenever A wants to communicate securely with B, the following happens:-


A
KA
KD
C
KB
B


A
A and B can now
communicate with
KS for encryption
B


A
Encrypted
with KA:
•Identities
of A & B.
•Random
number
(nonce)
Response
encrypted
with KA:
•KS
•A's request
•KS
encrypted
with KB
•A's ID
encrypted
with KB
KD
C

Security Handshake Pitfalls

One way authentication

Here, the authentication is only one way i.e. B authenticates A but A does not authenticate B.

Mutual Authentication

Both A and B authenticates each other.

One way authentication

Login only

•A sends login id & password in plain text.
•If it is valid, then communication starts.
•No further encryption or integrity checks are performed.

Shared secret

•A sends username to B.
•B creates a random challenge and sends to A.
•A encrypts random challenge with KAB

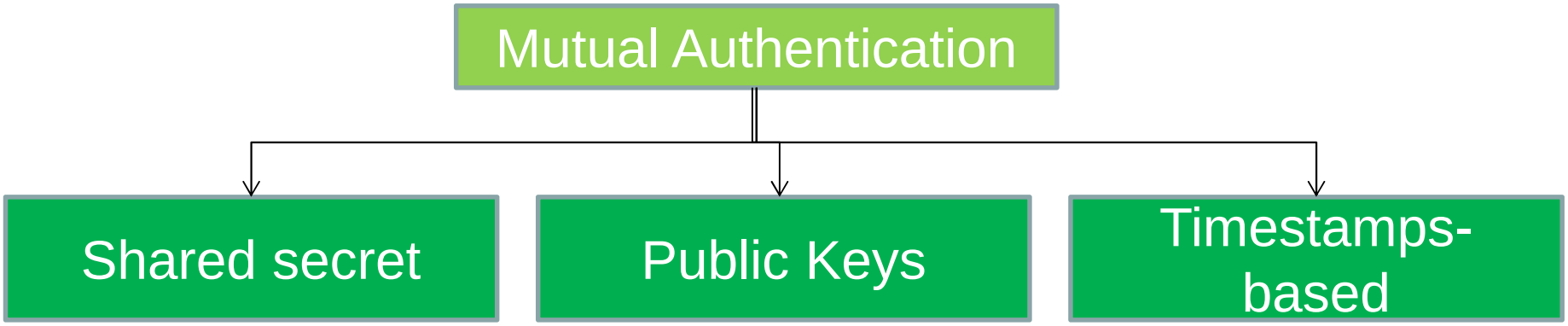
Modified approach:
•A sends username to B.
•B encrypts random challenge with KAB
•A then decrypts it and sends the original RC to B

One way public key

•A sends username to B.
•B creates a random challenge and sends to A.
•A encrypts random challenge with A's private key

Modified approach:
•A sends username to B.
•B encrypts random challenge with A's public key
•A then decrypts it with A's private key & sends the original RC to B

Mutual Authentication



Shared secret

•A sends username to B
•B sends a random challenge R1 to A
•R1 is encrypted with KAB
•A sends a random challenge R2 to B
•B encrypts R2 with KAB and sends it to A

Optimized Approach:
•A sends username and R2
•B sends encrypted R2 with KAB and R1
•A again sends encrypted R1 with KAB.

Public Keys

•A sends username and R2 encrypted with B's public key.
•B decrypts R2 with his private key. B creates a R1 and encrypts with A's public key
•A decrypts R1 and sends to B

Modified Approach:
•A sends username and R2
• B encrypts R2 with KAB & also sends R1.
•A then signs R1 and returns it back to B

Timestamps-based

•A sends username and current timestamp encrypted with KAB.
•B sends username and above timestamp +1 encrypted with KAB

Thank You