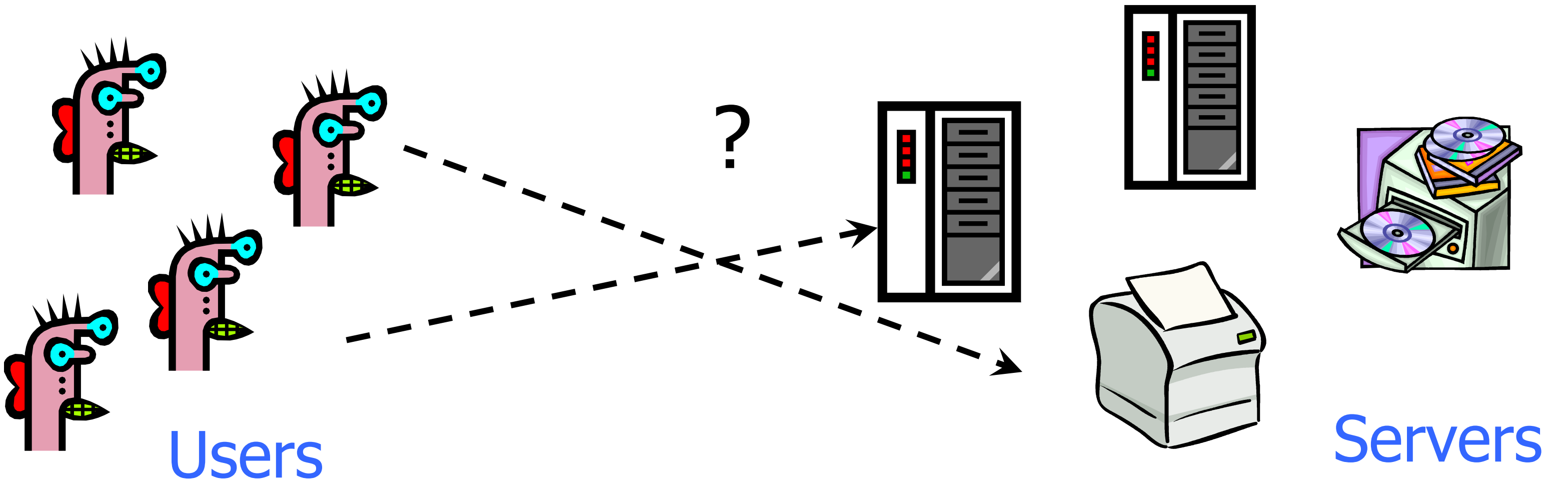
CS 5450

Authentication Protocols

Vitaly Shmatikov

Many-to-Many Authentication



How do users prove their identities when requesting services from machines on the network?

Naïve solution: every server knows every user's password

- Insecure: break into one server ⇒ compromise all users
- Inefficient: to change password, user must contact every server

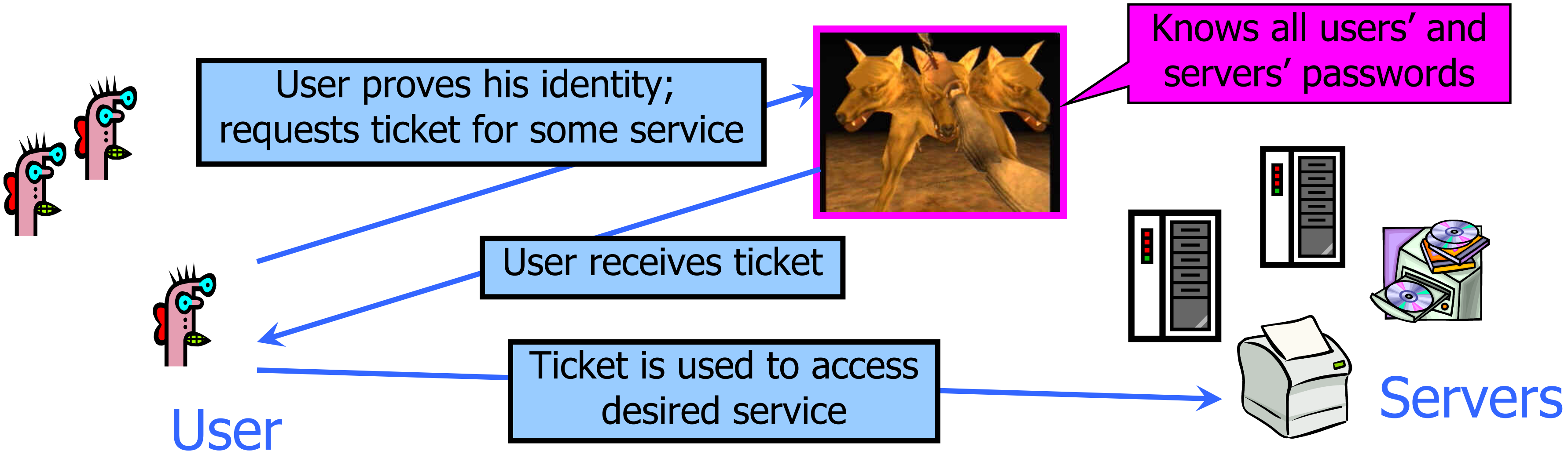
Requirements

- Security
  - … against attacks by passive eavesdroppers and actively malicious users
- Transparency
  - Users shouldn’t notice authentication taking place
  - Entering password is Ok, if done rarely
- Scalability
  - Large number of users and servers

Threats

- User impersonation
  - Malicious user with access to a workstation pretends to be another user from the same workstation
- Network address impersonation
  - Malicious user changes network address of his workstation to impersonate another workstation
- Eavesdropping, tampering, replay
  - Malicious user eavesdrops, tampers, or replays other users' conversations to gain unauthorized access

Solution: Trusted Third Party



User proves his identity; requests ticket for some service

Knows all users' and servers' passwords

User receives ticket

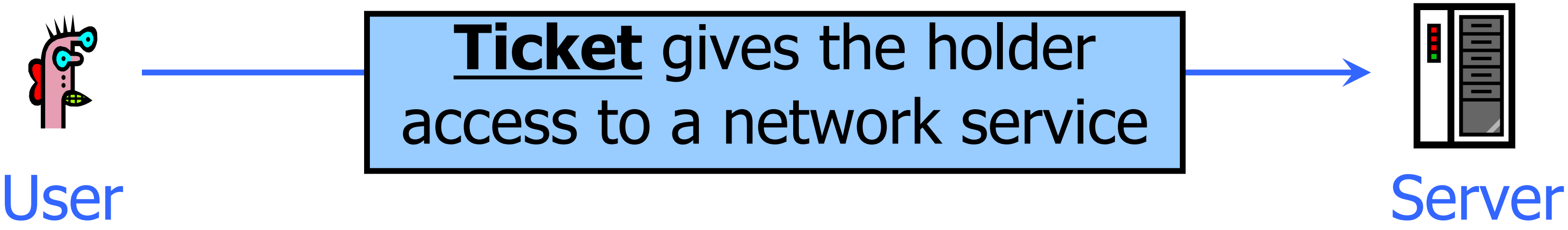
Ticket is used to access desired service

User

Servers

- Trusted authentication service on the network
  - Knows all passwords, can grant access to any server
  - Convenient (but also the single point of failure!)
  - Requires high level of physical security

What Should a Ticket Look Like?



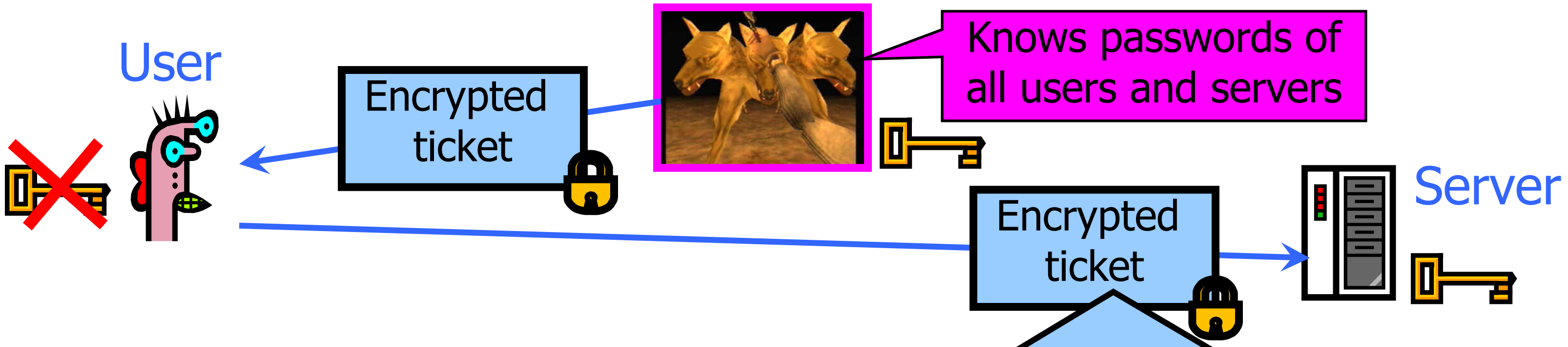
User

<u>Ticket</u> gives the holder access to a network service

Server

- User should not be able to access server without first proving his identity to authentication service
- Ticket proves that user has authenticated
  - Authentication service encrypts some information with a key known to the server (but not the user!)
    - The only thing the user can do is pass the ticket to the server
    - Hash functions would’ve worked well, but this is 1980s design
  - Server decrypts the ticket and verifies information

What Should a Ticket Include?


User

Encrypted ticket

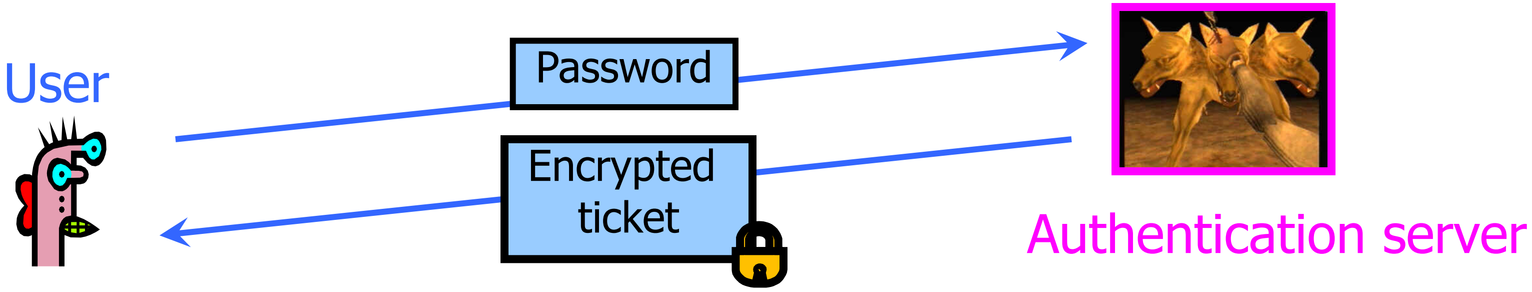
Knows passwords of all users and servers

Encrypted ticket

Server

- User name
- Server name
- Address of user’s workstation
  - Otherwise, a user on another workstation can steal the ticket and use it to gain access to the server
- Ticket lifetime
- A few other things (session key, etc.)

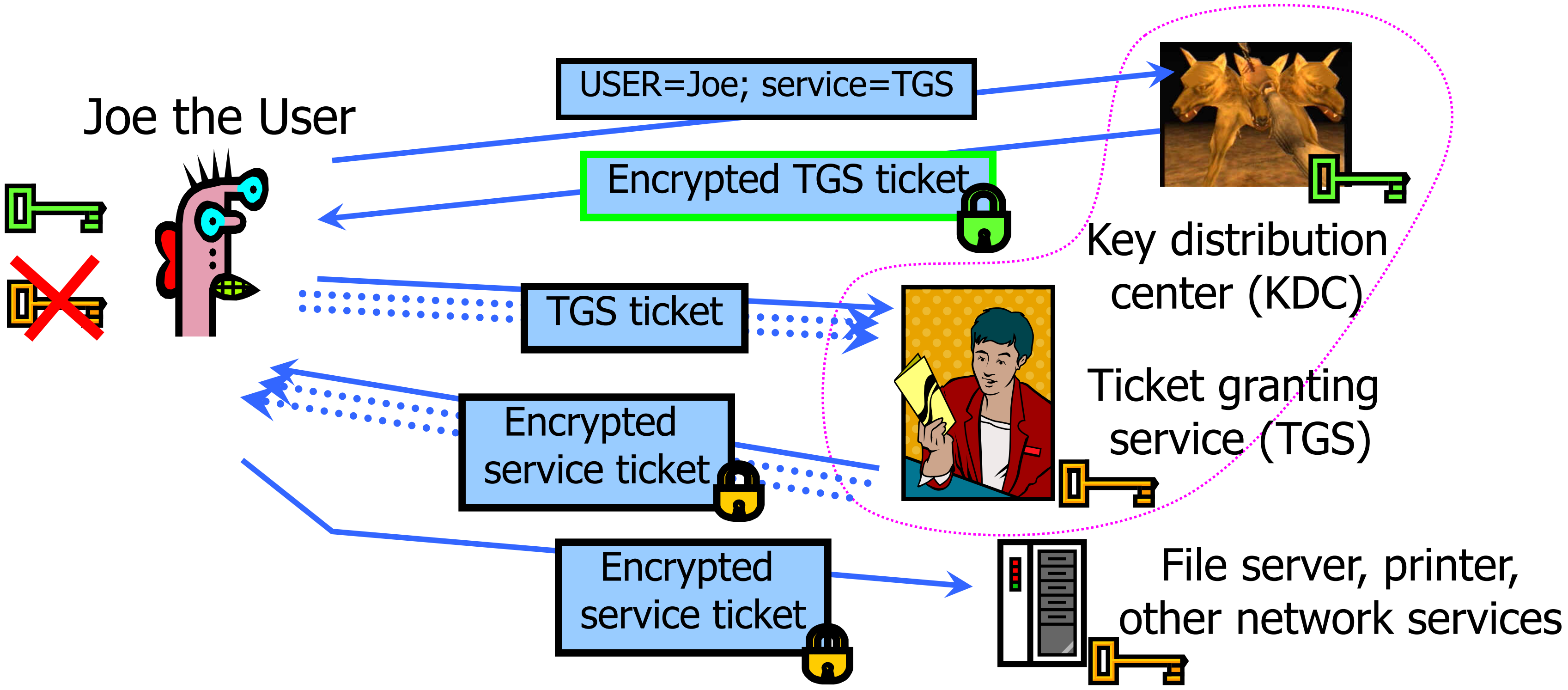
Naïve Authentication



- **Insecure:** passwords are sent in plaintext
  - Eavesdropper can steal the password and later impersonate the user to the authentication server
- **Inconvenient:** need to send the password each time to obtain the ticket for any network service
  - Separate authentication for email, printing, etc.

Two-Step Authentication

- Prove identity once to obtain a special TGS ticket
- Use TGS to get tickets for any network service



Joe the User

USER=Joe; service=TGS

Encrypted TGS ticket

TGS ticket

Encrypted service ticket

Encrypted service ticket

Key distribution center (KDC)

Ticket granting service (TGS)

File server, printer, other network services

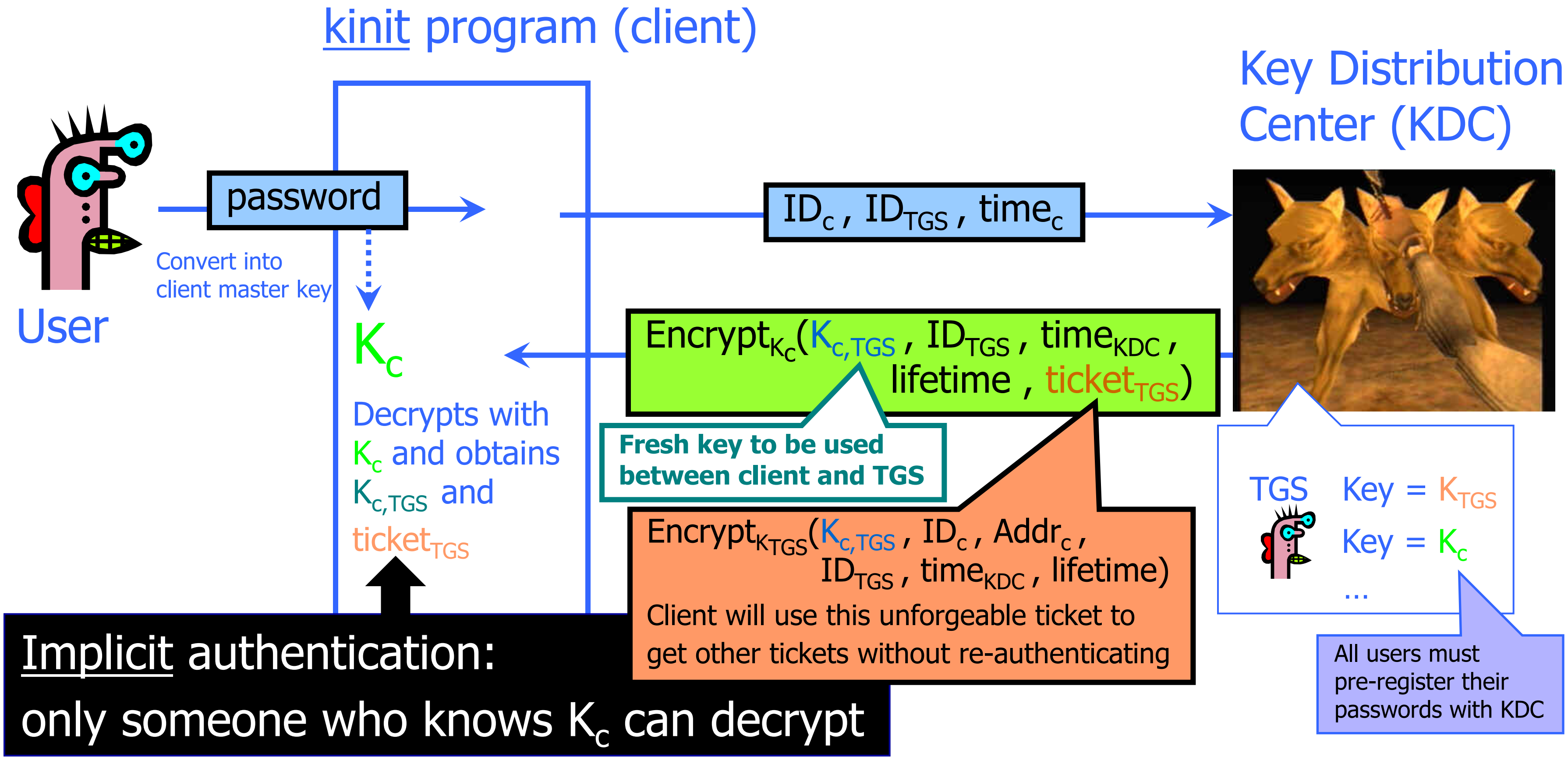
Threats

- Ticket hijacking
  - Malicious user may steal the service ticket of another user on the same workstation and try to use it
    - Network address verification does not help
  - Servers must verify that the user who is presenting the ticket is the same user to whom the ticket was issued
- No server authentication
  - Attacker may misconfigure the network so that he receives messages addressed to a legitimate server
    - Capture private information from users and/or deny service
  - Servers must prove their identity to users

Symmetric Keys in Kerberos

- K_c is long-term key of client C
  - Derived from the user’s password
  - Known to the client and the key distribution center (KDC)
- K_{TGS} is long-term key of TGS
  - Known to KDC and the ticket granting service (TGS)
- K_v is long-term key of network service V
  - Known to V and TGS; each service V has its own long-term key
- $K_{c,TGS}$ is short-term session key betw. C and TGS
  - Created by KDC, known to C and TGS
- $K_{c,v}$ is short-term session key between C and V
  - Created by TGS, known to C and V

“Single Logon” Authentication



kinit program (client)

Key Distribution Center (KDC)

User

password

Convert into client master key

K_c

Decrypts with K_c and obtains $K_{c,TGS}$ and $ticket_{TGS}$

ID_c , ID_{TGS} , $time_c$

$Encrypt_{K_c}(K_{c,TGS}$, ID_{TGS} , $time_{KDC}$, lifetime , $ticket_{TGS})$

Fresh key to be used between client and TGS

$Encrypt_{K_{TGS}}(K_{c,TGS}$, ID_c , $Addr_c$, ID_{TGS} , $time_{KDC}$, lifetime)

Client will use this unforgeable ticket to get other tickets without re-authenticating

TGS Key = K_{TGS}

Key = K_c

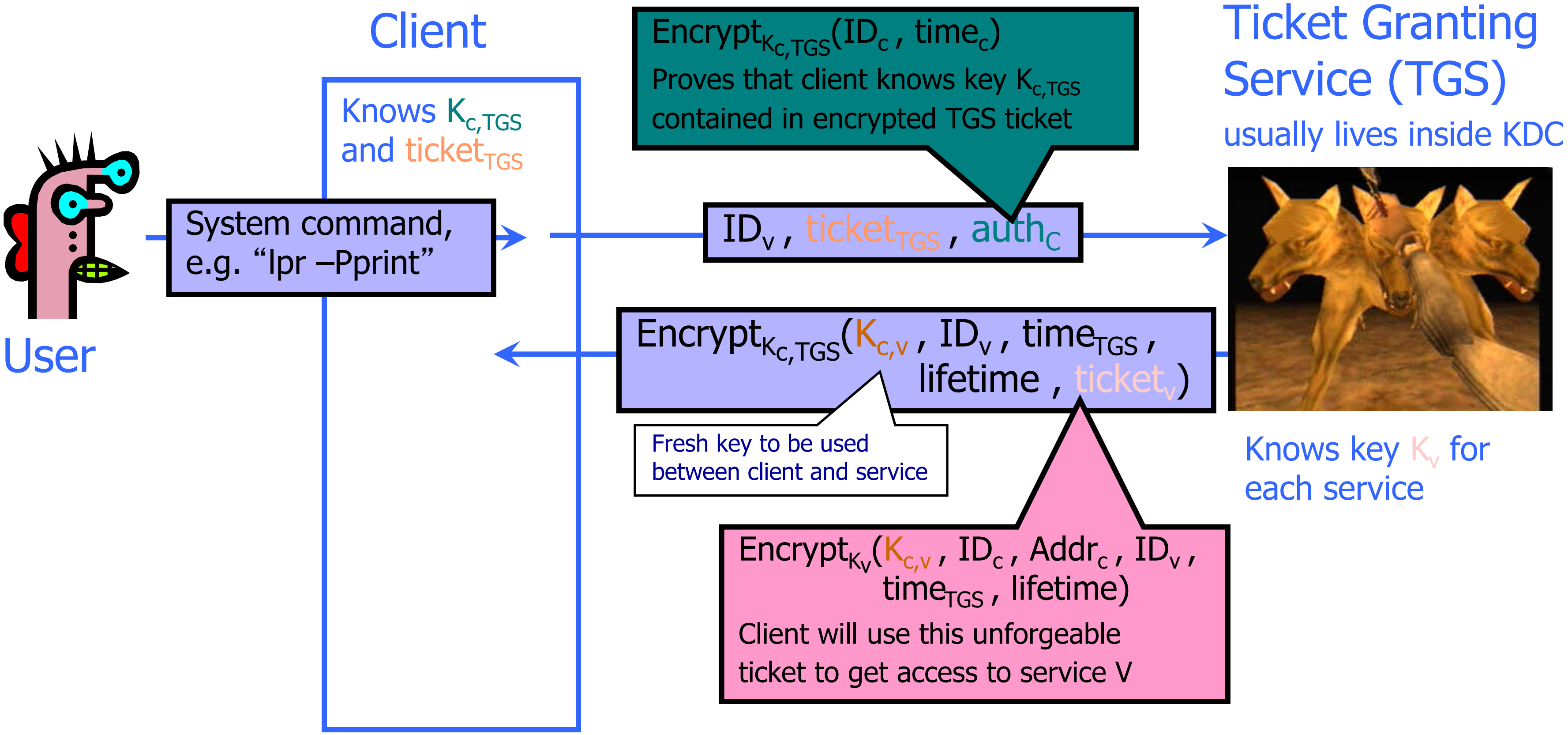
…

All users must pre-register their passwords with KDC

Implicit authentication:
only someone who knows K_c can decrypt

- Client only needs to obtain TGS ticket **once** (say, every morning)
- Ticket is encrypted; client cannot forge it or tamper with it

Obtaining a Service Ticket



Client

Knows $K_{c,TGS}$ and $ticket_{TGS}$

User

System command, e.g. “lpr –Pprint”

ID_v , $ticket_{TGS}$, $auth_C$

$Encrypt_{K_{c,TGS}}(ID_c , time_c)$

Proves that client knows key $K_{c,TGS}$ contained in encrypted TGS ticket

Ticket Granting Service (TGS)

usually lives inside KDC

$Encrypt_{K_{c,TGS}}(K_{c,v} , ID_v , time_{TGS} , lifetime , ticket_v)$

Fresh key to be used between client and service

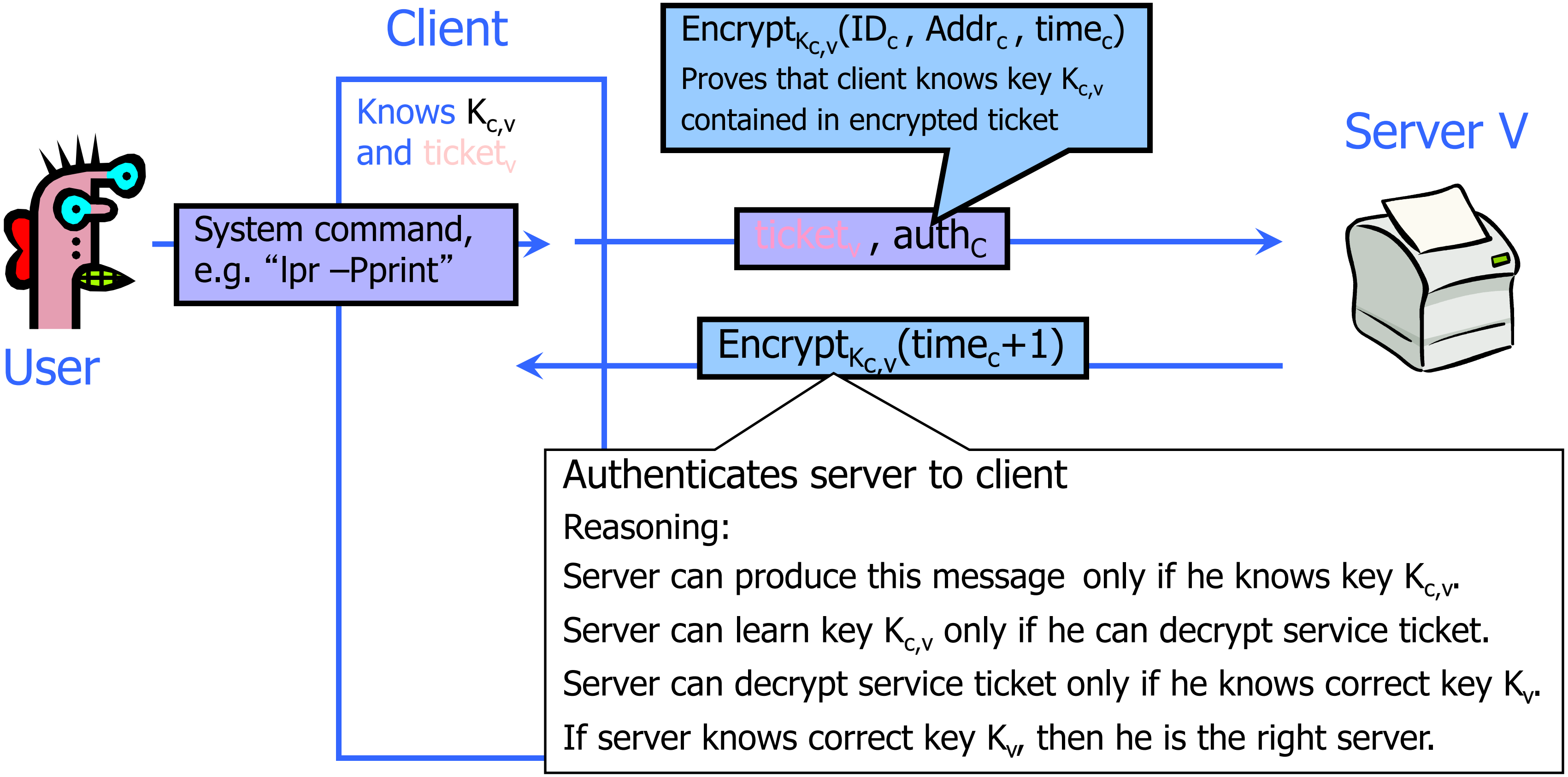
Knows key K_v for each service

$Encrypt_{K_v}(K_{c,v} , ID_c , Addr_c , ID_v , time_{TGS} , lifetime)$

Client will use this unforgeable ticket to get access to service V

- Client uses TGS ticket to obtain a service ticket and a short-term session key for each network service (printer, email, etc.)

Obtaining Service



Client

Knows $K_{c,v}$ and ticket$_v$

User

System command, e.g. “lpr –Pprint”

Encrypt$_{K_{c,v}}$(ID$_c$, Addr$_c$, time$_c$)

Proves that client knows key $K_{c,v}$ contained in encrypted ticket

ticket$_v$, auth$_C$

Server V

Encrypt$_{K_{c,v}}$(time$_c$+1)

Authenticates server to client

Reasoning:

Server can produce this message only if he knows key $K_{c,v}$.

Server can learn key $K_{c,v}$ only if he can decrypt service ticket.

Server can decrypt service ticket only if he knows correct key K_v.

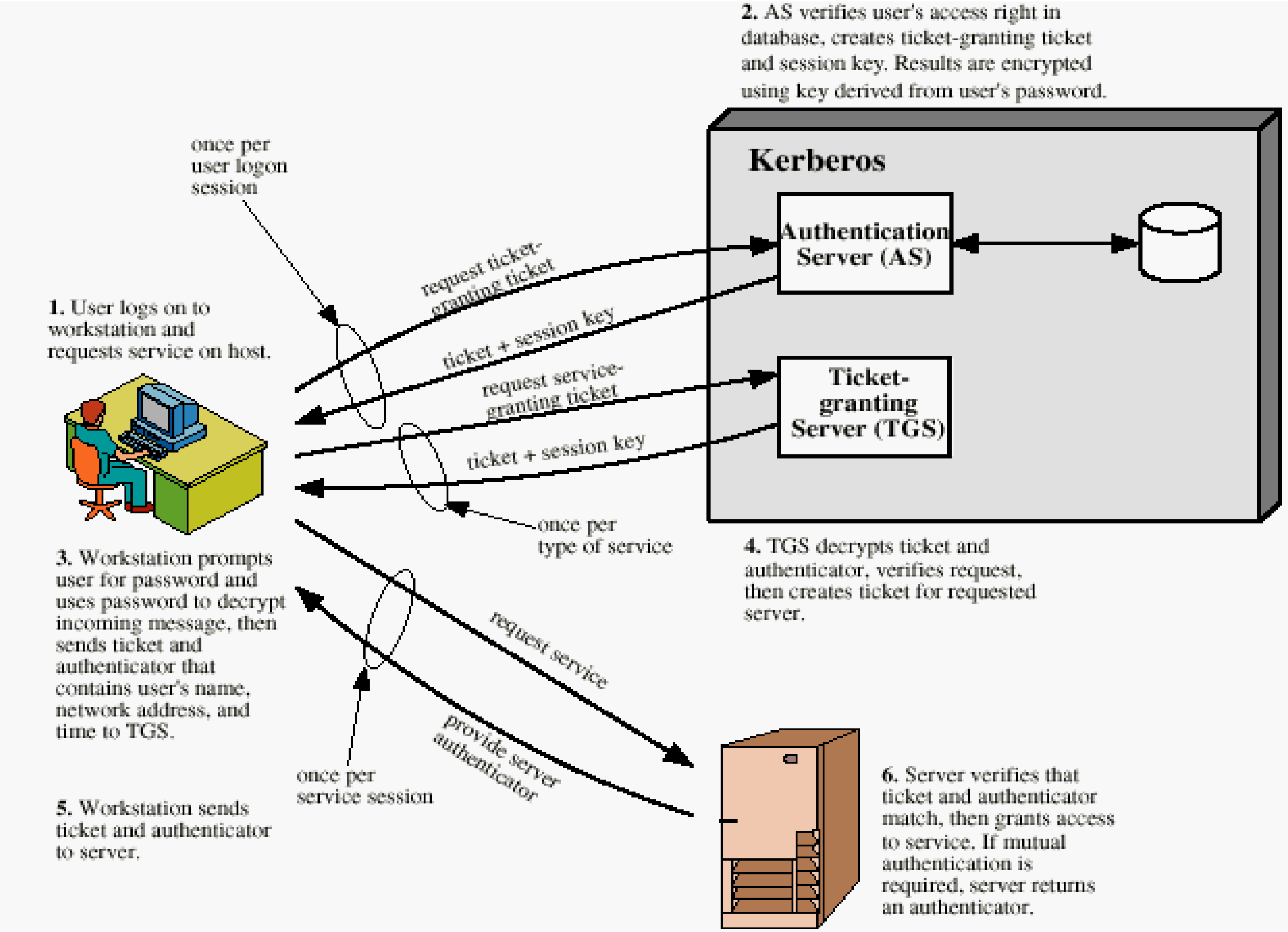
If server knows correct key K_v, then he is the right server.

- For each service request, client uses the short-term key for that service and the ticket he received from TGS

Kerberos in Large Networks

- One KDC isn’t enough for large networks (why?)
- Network is divided into realms
  - KDCs in different realms have different key databases
- To access a service in another realm, users must…
  - Get ticket for home-realm TGS from home-realm KDC
  - Get ticket for remote-realm TGS from home-realm TGS
    - As if remote-realm TGS were just another network service
  - Get ticket for remote service from that realm’s TGS
  - Use remote-realm ticket to access service
  - N(N-1)/2 key exchanges for full N-realm interoperation

Summary of Kerberos



1. User logs on to workstation and requests service on host.

2. AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.

3. Workstation prompts user for password and uses password to decrypt incoming message, then sends ticket and authenticator that contains user's name, network address, and time to TGS.

4. TGS decrypts ticket and authenticator, verifies request, then creates ticket for requested server.

5. Workstation sends ticket and authenticator to server.

6. Server verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.

Kerberos
Authentication Server (AS)
Ticket-granting Server (TGS)
once per user logon session
request ticket-granting ticket
ticket + session key
request service-granting ticket
ticket + session key
once per type of service
request service
provide server authenticator
once per service session

Important Ideas in Kerberos

- Short-term session keys
  - Long-term secrets used only to derive short-term keys
  - Separate session key for each user-server pair
    - Re-used by multiple sessions between same user and server
- Proofs of identity based on authenticators
  - Client encrypts his identity, addr, time with session key; knowledge of key proves client has authenticated to KDC
    - Also prevents replays (if clocks are globally synchronized)
  - Server learns this key separately (via encrypted ticket that client can’t decrypt), verifies client’s authenticator
- Symmetric cryptography only

Kerberos Version 5

- Better user-server authentication
  - Separate subkey for each user-server session instead of re-using the session key contained in the ticket
  - Authentication via subkeys, not timestamp increments
- Authentication forwarding (delegation)
  - Servers can access other servers on user's behalf, eg, can tell printer to fetch email
- Realm hierarchies for inter-realm authentication
- Explicit integrity checking + standard CBC mode
- Multiple encryption schemes, not just DES

Practical Uses of Kerberos

- Microsoft Windows
- Email, FTP, network file systems, many other applications have been kerberized
  - Use of Kerberos is transparent for the end user
  - Transparency is important for usability!
- Local authentication
  - login and su in OpenBSD
- Authentication for network protocols
  - rlogin, rsh
- Secure windowing systems

What Is SSL / TLS?

- Secure Sockets Layer and Transport Layer Security protocols
  - Same protocol design, different crypto algorithms
- De facto standard for Internet security
  - “The primary goal of the TLS protocol is to provide privacy and data integrity between two communicating applications”
- Deployed in every Web browser; also VoIP, payment systems, distributed systems, etc.

SSL / TLS Guarantees

- End-to-end secure communications in the presence of a network attacker
  - Attacker completely 0wns the network: controls Wi-Fi, DNS, routers, his own websites, can listen to any packet, modify packets in transit, inject his own packets into the network
- Scenario: you are reading your email from an Internet café connected via a r00ted Wi-Fi access point to a dodgy ISP in a hostile authoritarian country

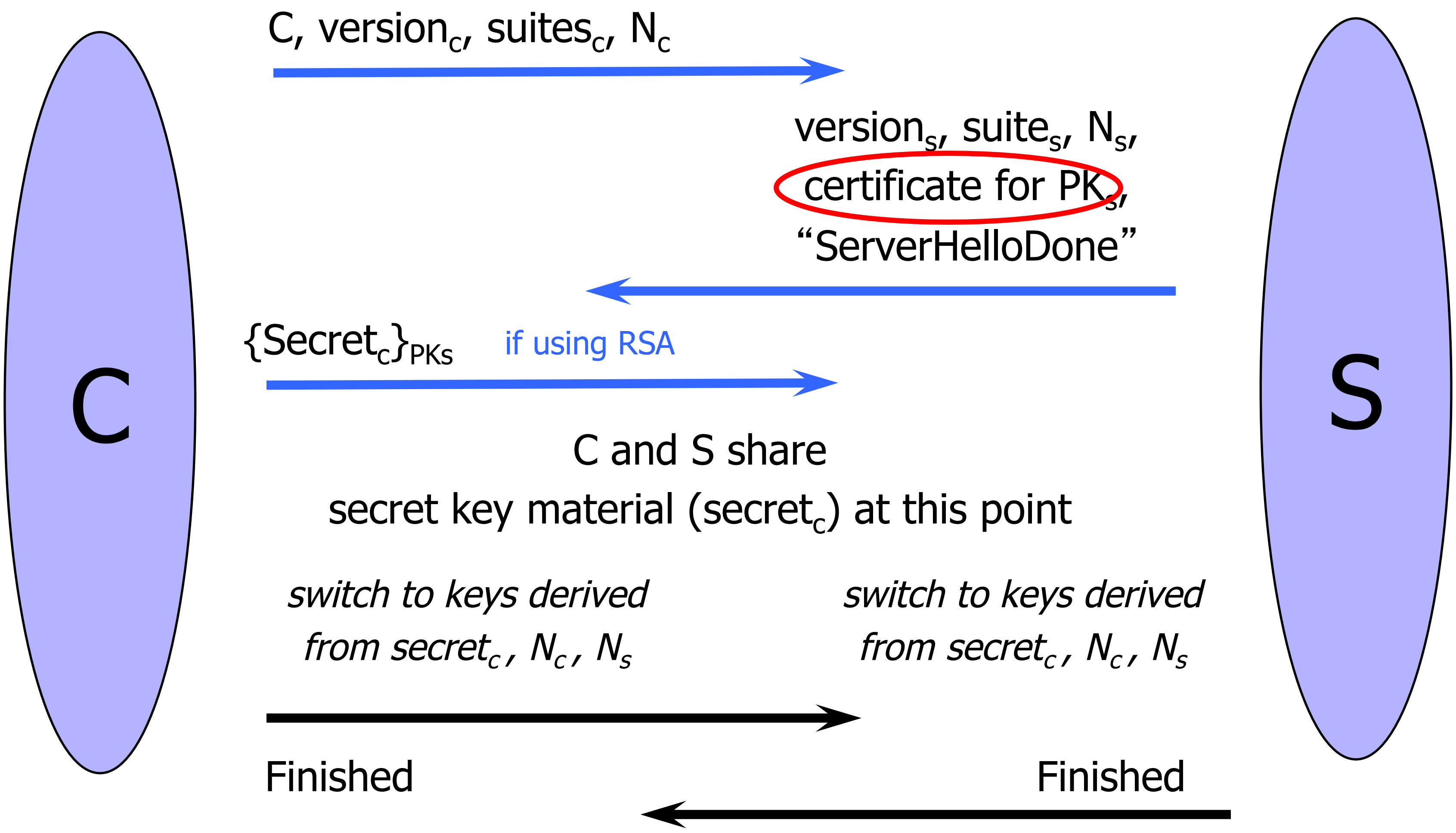
SSL Basics: Two Protocols

- Handshake protocol
  - Uses public-key cryptography to establish several shared secret keys between the client and the server
- Record protocol
  - Uses the secret keys established in the handshake protocol to protect confidentiality, integrity, and authenticity of data exchange between the client and the server

SSL Handshake Protocol

- Runs between a client and a server
  - For example, client = Web browser, server = website
- Negotiate version of the protocol and the set of cryptographic algorithms to be used
  - Interoperability between different implementations
- Authenticate server and client (optional)
  - Use digital certificates to learn each other's public keys and verify each other's identity
  - Often only the server is authenticated
- Use public keys to establish a shared secret

“Core” SSL Handshake



C

S

C, version_c, suites_c, N_c

version_s, suite_s, N_s,

certificate for PK_s,

“ServerHelloDone”

$\{\text{Secret}_c\}_{PKs}$ if using RSA

C and S share

secret key material (secret_c) at this point

switch to keys derived from secret_c, N_c, N_s

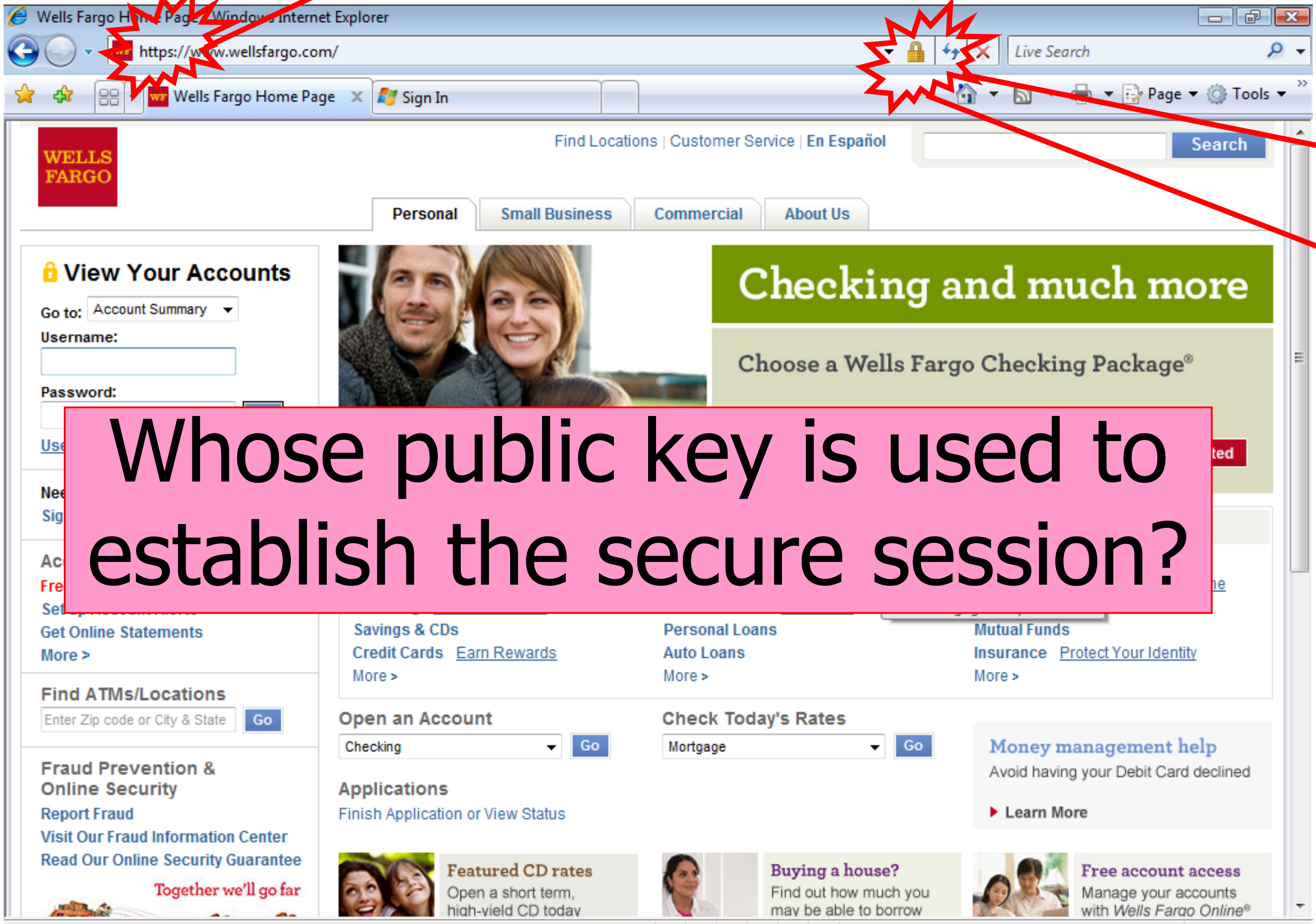
switch to keys derived from secret_c, N_c, N_s

Finished

Finished

Motivation

https://

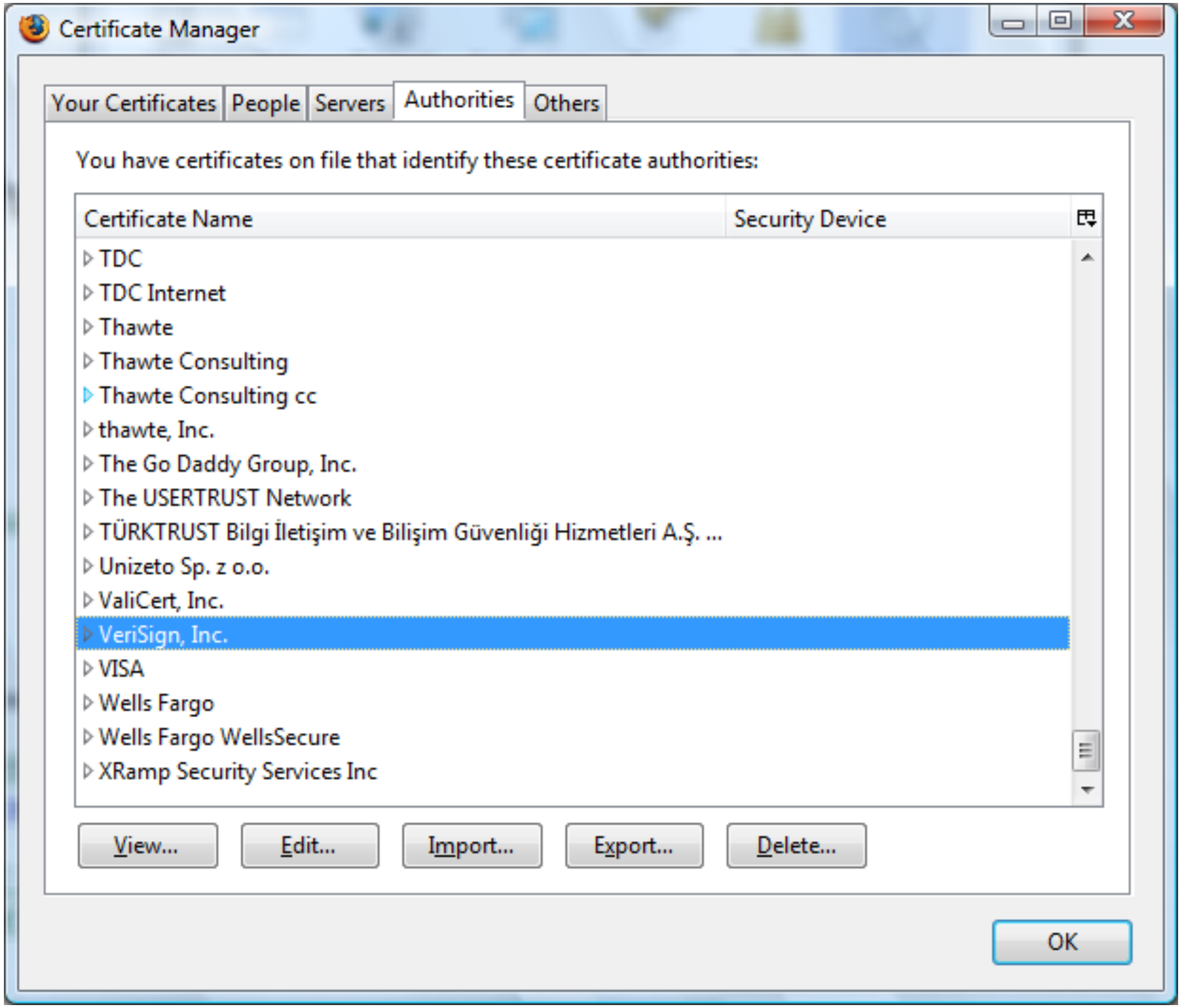

Wells Fargo Home Page - Windows Internet Explorer
https://www.wellsfargo.com/
Live Search
Wells Fargo Home Page
Sign In
Page
Tools
WELLS FARGO
Find Locations | Customer Service | En Español
Search
Personal
Small Business
Commercial
About Us
View Your Accounts
Go to: Account Summary
Username:
Password:
Checking and much more
Choose a Wells Fargo Checking Package®
Get Online Statements
More >
Savings & CDs
Credit Cards Earn Rewards
More >
Personal Loans
Auto Loans
More >
Mutual Funds
Insurance Protect Your Identity
More >
Find ATMs/Locations
Enter Zip code or City & State
Go
Open an Account
Checking
Go
Check Today's Rates
Mortgage
Go
Money management help
Avoid having your Debit Card declined
Learn More
Applications
Finish Application or View Status
Fraud Prevention & Online Security
Report Fraud
Visit Our Fraud Information Center
Read Our Online Security Guarantee
Together we'll go far
Featured CD rates
Open a short term, high-yield CD today
Buying a house?
Find out how much you may be able to borrow
Free account access
Manage your accounts with Wells Fargo Online®

Whose public key is used to establish the secure session?

Distribution of Public Keys

- Public-key certificate
  - Signed statement specifying the key and identity
    - $\text{sig}_{\text{Alice}}(\text{“Bob”}, \text{PK}_B)$
- Common approach: certificate authority (CA)
  - An agency responsible for certifying public keys
  - Browsers are pre-configured with 100+ of trusted CAs
  - A public key for any website in the world will be accepted by the browser if certified by one of these CAs

Trusted Certificate Authorities

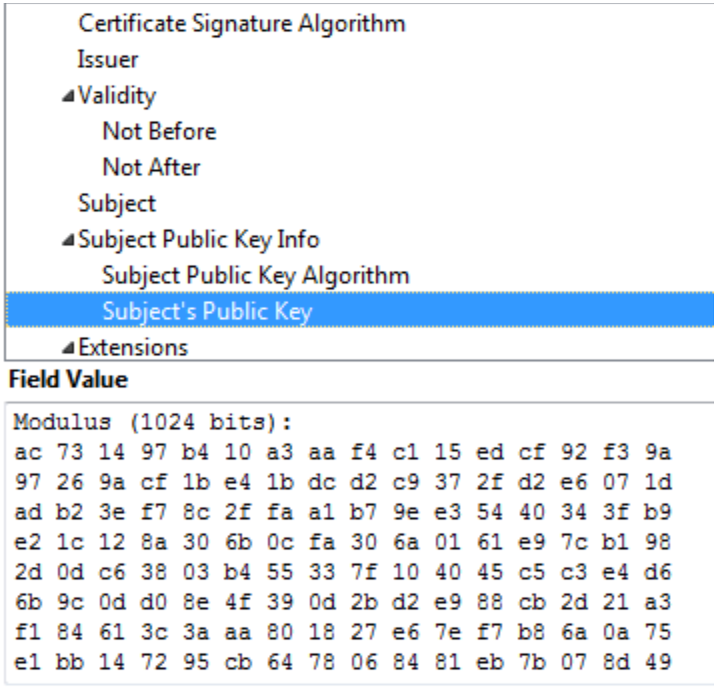


You have certificates on file that identify these certificate authorities:

Certificate Name	Security Device
TDC	
TDC Internet	
Thawte	
Thawte Consulting	
Thawte Consulting cc	
thawte, Inc.	
The Go Daddy Group, Inc.	
The USERTRUST Network	
TÜRKTRUST Bilgi İletişim ve Bilişim Güvenliği Hizmetleri A.Ş. ...	
Unizeto Sp. z o.o.	
ValiCert, Inc.	
VeriSign, Inc.	
VISA	
Wells Fargo	
Wells Fargo WellsSecure	
XRamp Security Services Inc	

Example of a Certificate

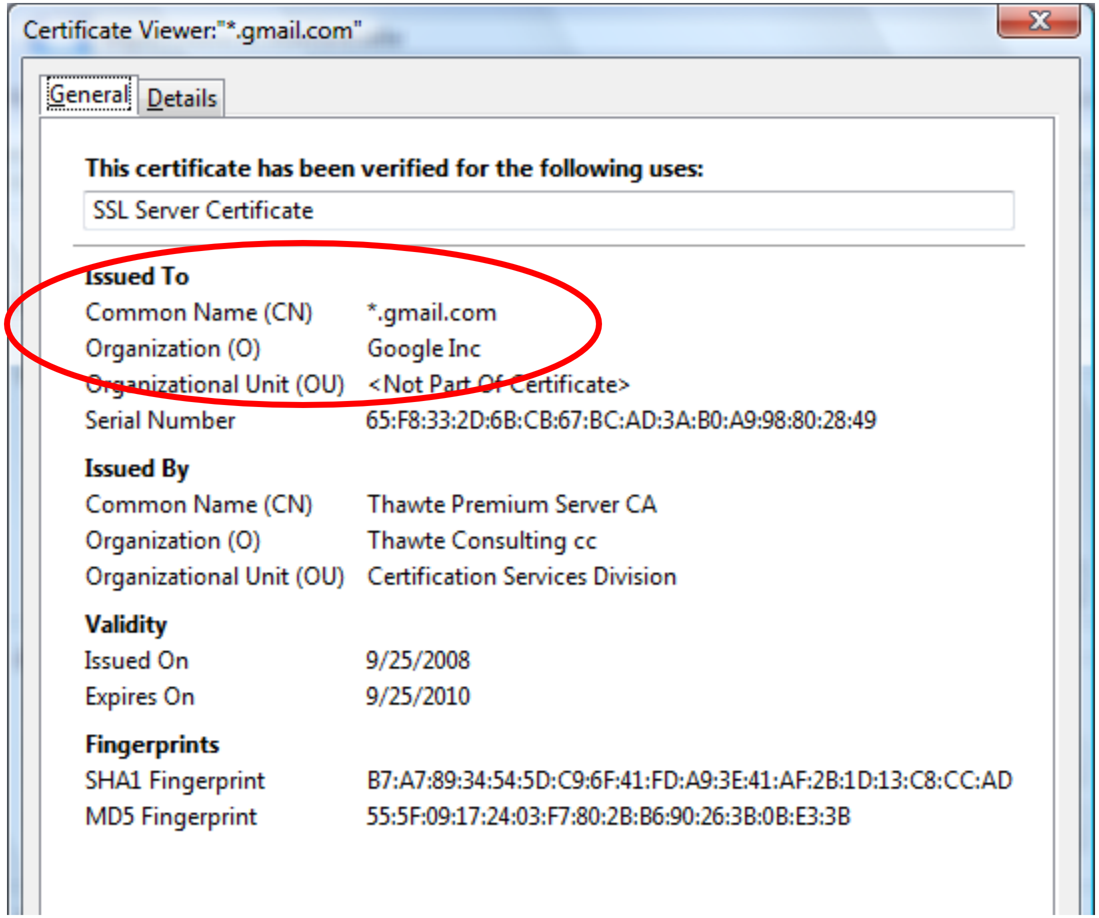
Important fields



- Certificate Signature Algorithm
- Issuer
- ◢Validity
  - Not Before
  - Not After
- Subject
- ◢Subject Public Key Info
  - Subject Public Key Algorithm
  - Subject's Public Key
- ◢Extensions

Field Value

```
Modulus (1024 bits):
ac 73 14 97 b4 10 a3 aa f4 c1 15 ed cf 92 f3 9a
97 26 9a cf 1b e4 1b dc d2 c9 37 2f d2 e6 07 1d
ad b2 3e f7 8c 2f fa a1 b7 9e e3 54 40 34 3f b9
e2 1c 12 8a 30 6b 0c fa 30 6a 01 61 e9 7c b1 98
2d 0d c6 38 03 b4 55 33 7f 10 40 45 c5 c3 e4 d6
6b 9c 0d d0 8e 4f 39 0d 2b d2 e9 88 cb 2d 21 a3
f1 84 61 3c 3a aa 80 18 27 e6 7e f7 b8 6a 0a 75
e1 bb 14 72 95 cb 64 78 06 84 81 eb 7b 07 8d 49
```



This certificate has been verified for the following uses:

SSL Server Certificate

Issued To

Common Name (CN)	*.gmail.com
Organization (O)	Google Inc
Organizational Unit (OU)	<Not Part Of Certificate>
Serial Number	65:F8:33:2D:6B:CB:67:BC:AD:3A:B0:A9:98:80:28:49

Issued By

Common Name (CN)	Thawte Premium Server CA
Organization (O)	Thawte Consulting cc
Organizational Unit (OU)	Certification Services Division

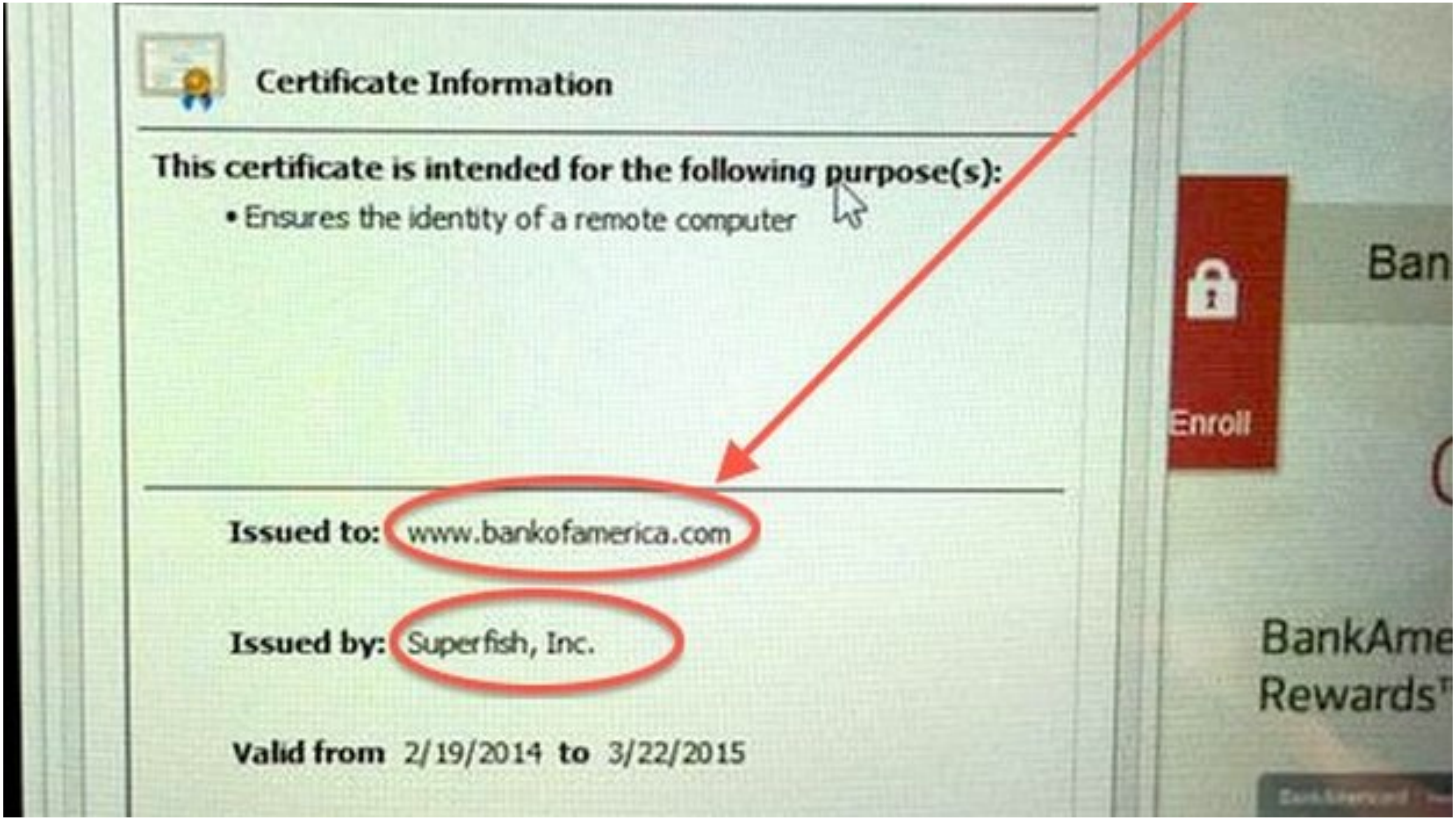
Validity

Issued On	9/25/2008
Expires On	9/25/2010

Fingerprints

SHA1 Fingerprint	B7:A7:89:34:54:5D:C9:6F:41:FD:A9:3E:41:AF:2B:1D:13:C8:CC:AD
MD5 Fingerprint	55:5F:09:17:24:03:F7:80:2B:B6:90:26:3B:0B:E3:3B

Another Example of a Certificate



Certificate Information

This certificate is intended for the following purpose(s):

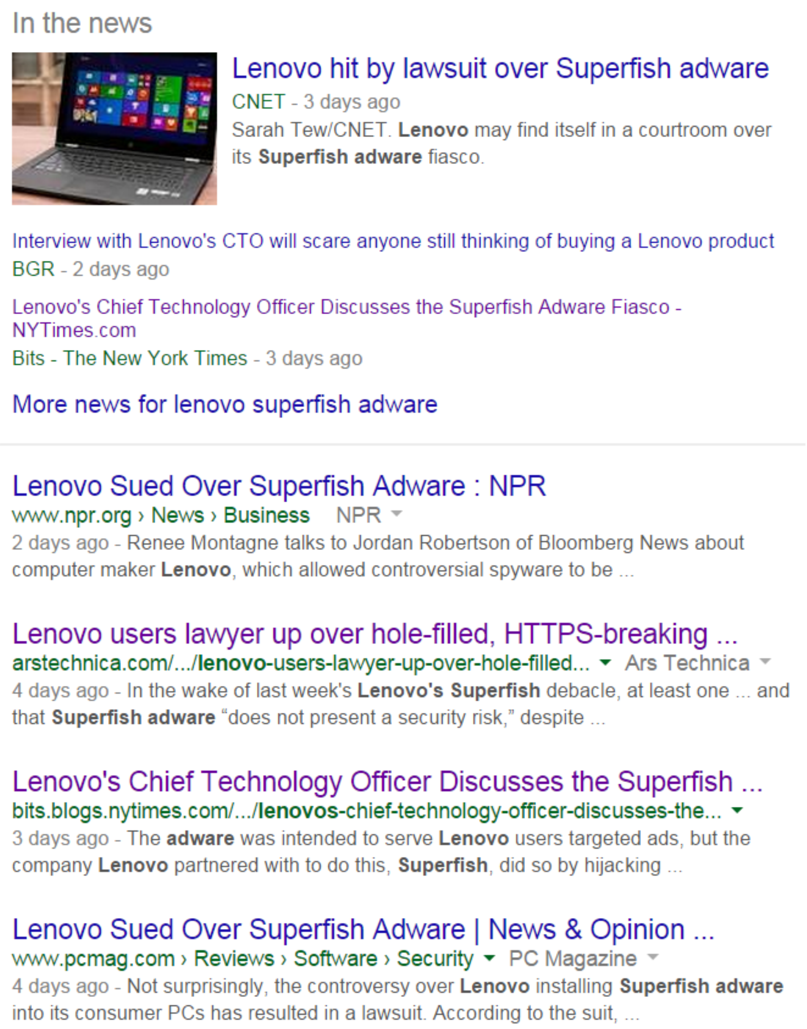
- Ensures the identity of a remote computer

Issued to: www.bankofamerica.com

Issued by: Superfish, Inc.

Valid from 2/19/2014 to 3/22/2015

Root Certificates in Lenovo



In the news

Lenovo hit by lawsuit over Superfish adware
CNET - 3 days ago
Sarah Tew/CNET. **Lenovo** may find itself in a courtroom over its **Superfish adware** fiasco.

Interview with Lenovo's CTO will scare anyone still thinking of buying a Lenovo product
BGR - 2 days ago

Lenovo's Chief Technology Officer Discusses the Superfish Adware Fiasco - NYTimes.com
Bits - The New York Times - 3 days ago

More news for lenovo superfish adware

Lenovo Sued Over Superfish Adware : NPR
www.npr.org › News › Business NPR
2 days ago - Renee Montagne talks to Jordan Robertson of Bloomberg News about computer maker **Lenovo**, which allowed controversial spyware to be ...

Lenovo users lawyer up over hole-filled, HTTPS-breaking ...
arstechnica.com/.../**lenovo-users-lawyer-up-over-hole-filled**... Ars Technica
4 days ago - In the wake of last week's **Lenovo's Superfish** debacle, at least one ... and that **Superfish adware** “does not present a security risk,” despite ...

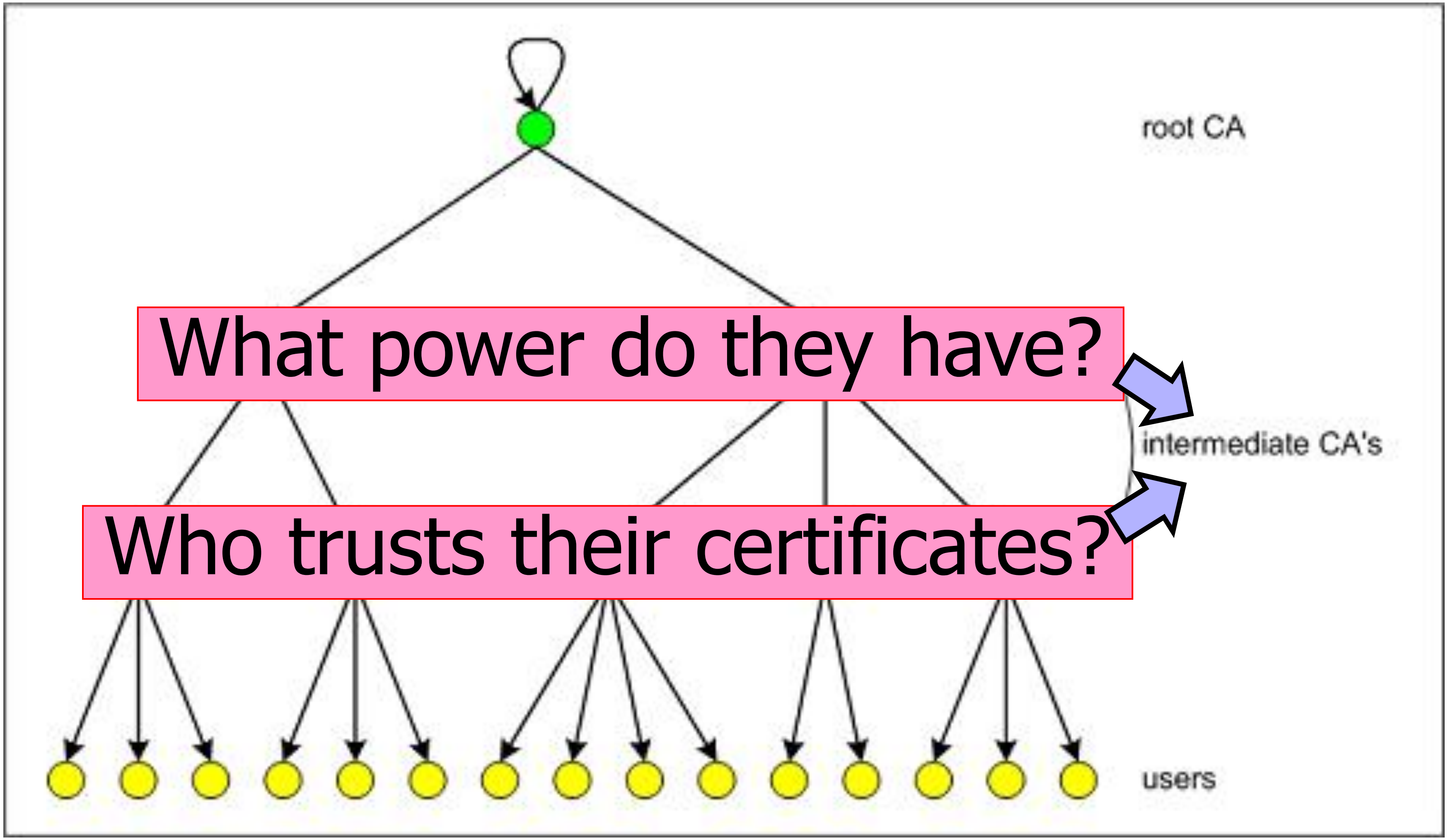
Lenovo's Chief Technology Officer Discusses the Superfish ...
bits.blogs.nytimes.com/.../**lenovos-chief-technology-officer-discusses-the**...
3 days ago - The **adware** was intended to serve **Lenovo** users targeted ads, but the company **Lenovo** partnered with to do this, **Superfish**, did so by hijacking ...

Lenovo Sued Over Superfish Adware | News & Opinion ...
www.pcmag.com › Reviews › Software › Security PC Magazine
4 days ago - Not surprisingly, the controversy over **Lenovo** installing **Superfish adware** into its consumer PCs has resulted in a lawsuit. According to the suit, ...

CA Hierarchy

- Browsers, operating systems, etc. have trusted root certificate authorities
  - My Chrome includes certificates of 195 trusted root CAs
- A Root CA signs certificates for intermediate CAs, they sign certificates for lower-level CAs, etc.
  - Certificate "chain of trust"
    - $\mathrm{sig}_{\mathrm{Verisign}}$("Cornell", $\mathrm{PK}_{\mathrm{Cornell}}$), $\mathrm{sig}_{\mathrm{Cornell}}$("Vitaly S.", $\mathrm{PK}_{\mathrm{Vitaly}}$)
- CA is responsible for verifying the identities of certificate requestors, domain ownership

Certificate Hierarchy



What power do they have?

Who trusts their certificates?

Flame

- Cyber-espionage virus (2010-2012)
- Signed with a fake intermediate CA certificate accepted by any Windows Update service
  - … created using an MD5 chosen-prefix collision against an obscure Microsoft Terminal Server Licensing Service certificate that was enabled for code signing and still used MD5 hash function
  - Pre-dates public knowledge of hash collisions in MD5
    - Evidence of state-level cryptanalysis?

TurkTrust

- In Jan 2013, a rogue *.google.com certificate was issued by an intermediate CA that gained its authority from the Turkish root CA TurkTrust
  - TurkTrust accidentally issued intermediate CA certs to customers who requested regular certificates
  - Ankara transit authority used its certificate to issue a fake *.google.com certificate in order to intercept and filter SSL traffic from its network
- This rogue *.google.com certificate was trusted by every browser in the world

TrustWave

Trustwave®
Information Security & Compliance

- In Feb 2012, admitted issuing an intermediate CA certificate to a corporate customer
  - Purpose: "re-sign" certificates for "data loss prevention"
  - Translation: forge certificates of third-party sites in order to spy on employees' encrypted communications with the outside world
- Customer can now forge certificates for any site in world… and they will be accepted by any browser!
  - What if a "re-signed" certificate leaks out?
- Do other CAs do this?

Komodia

- Israeli startup
- From their website: “Our advanced SSL hijacker SDK is a brand new technology that allows you to access data that was encrypted using SSL and perform on the fly SSL decryption”
  - Installs its own root certificate
  - Goal: re-sign SSL certificates, proxy/MITM connections
- Same private key on all machines, easily extracted
  - Anyone can issue fake Komodia certificates, do man-in-the-middle attacks on any machine with Komodia

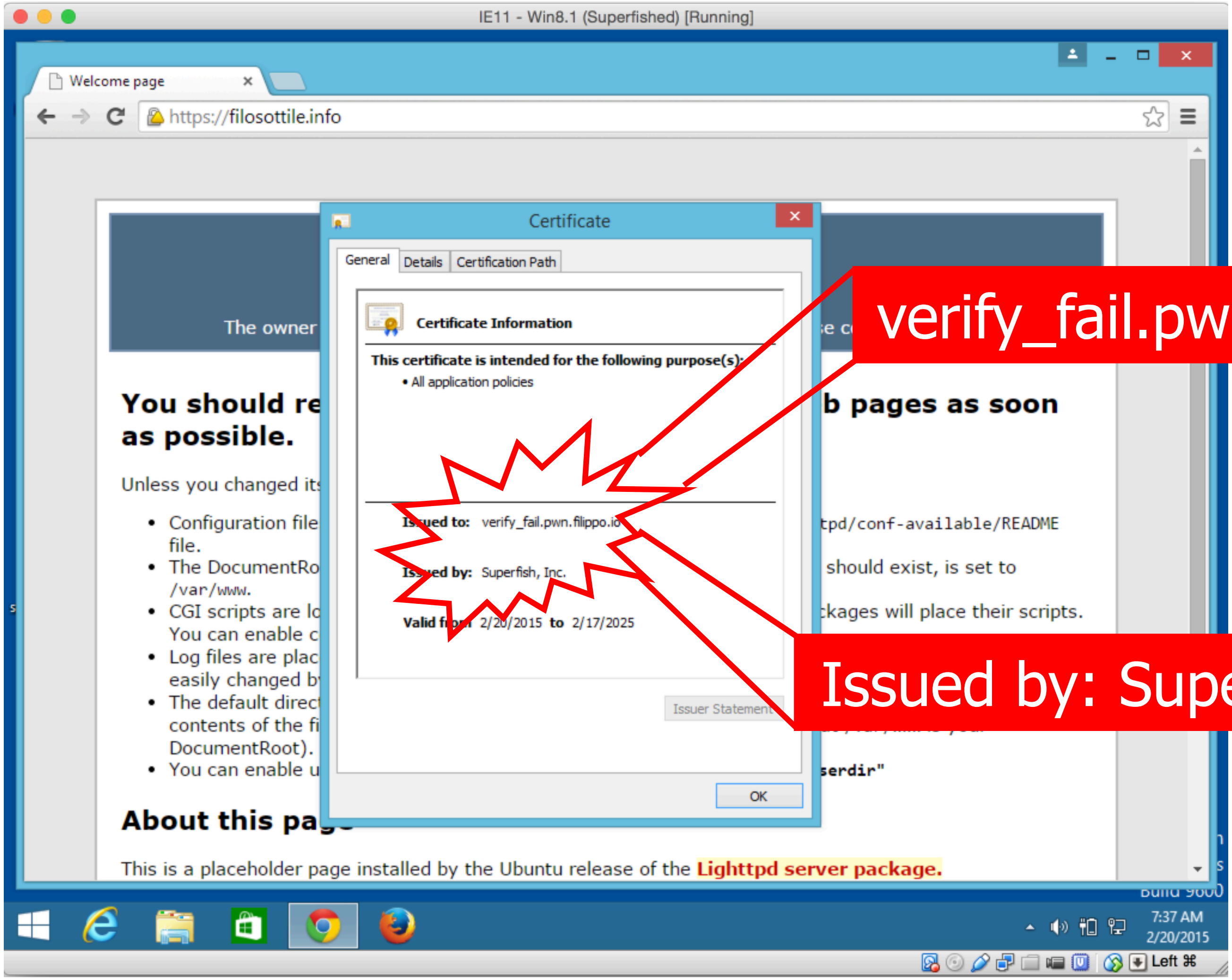
It Gets Worse

[redacted] https://blog.filippo.io/komodia-superfish-ssl-validation-is-broken/

- What happens if a MITM attacker serves a self-signed certificate to a Komodia client?
- Komodia re-signs and turns it into a trusted certificate
  - But it will also change the name in the certificate, which won’t match what the browser is expecting and user will see a warning - maybe not so bad
- But if attacker puts target domain into “alternate name” field, Komodia won’t touch it and browser will think the certificate is completely valid

Complete SSL Fail

[redacted] https://blog.filippo.io/komodia-superfish-ssl-validation-is-broken/



Certificate Information

This certificate is intended for the following purpose(s):

- All application policies

Issued to: verify_fail.pwn.filippo.io

Issued by: Superfish, Inc.

Valid from 2/20/2015 **to** 2/17/2025

The owner ... e c

You should re ... b pages as soon as possible.

Unless you changed its

- Configuration file ... tpd/conf-available/README file.
- The DocumentRo ... should exist, is set to /var/www.
- CGI scripts are lo ... You can enable c ... kages will place their scripts.
- Log files are plac ... easily changed b
- The default direct ... contents of the fi ... DocumentRoot).
- You can enable u ... serdir"

About this pa

This is a placeholder page installed by the Ubuntu release of the **Lighttpd server package.**

verify_fail.pwn.filippo.io

Issued by: Superfish, Inc.

Statement from Superfish CEO

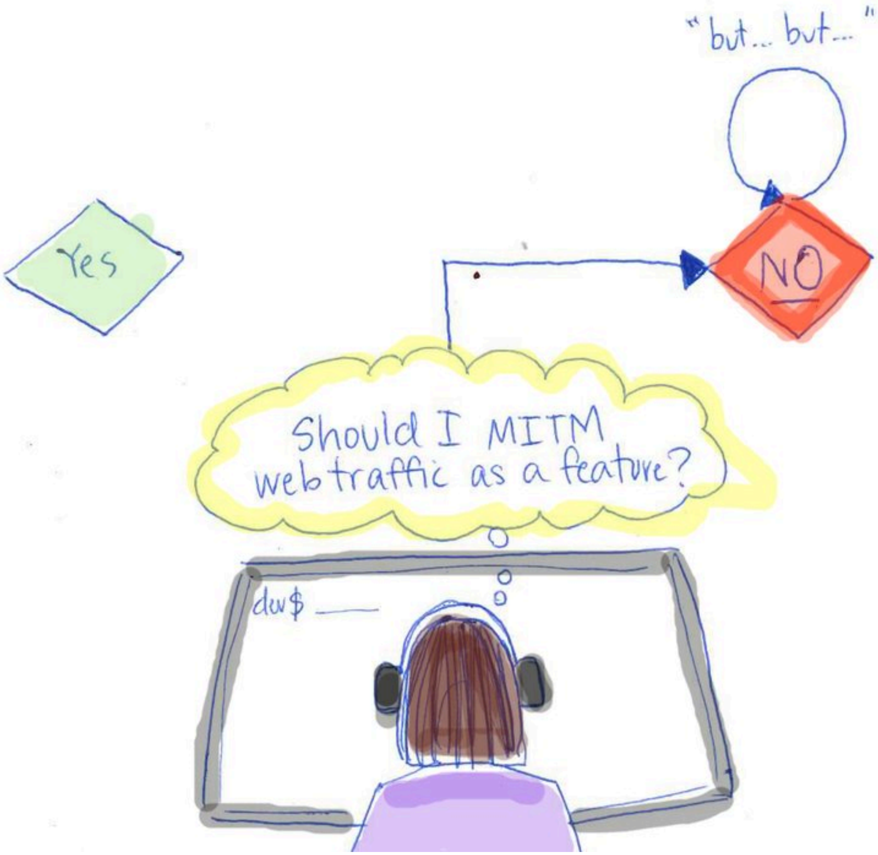
There has been significant misinformation circulating about Superfish software that was pre-installed on certain Lenovo laptops. The software shipped on a limited number of computers in 2014 in an effort to enhance the online shopping experience for Lenovo customers. Superfish's software utilizes visual search technology to help users achieve more relevant search results based on images of products they have browsed.

Despite the false and misleading statements made by some media commentators and bloggers, the Superfish software does not present a security risk. In no way does Superfish store personal data or share such data with anyone. Unfortunately, in this situation a vulnerability was introduced unintentionally by a 3rd party. Both Lenovo and Superfish did extensive testing of the solution but this issue wasn't identified before some laptops shipped. Fortunately, our partnership with Lenovo was limited in scale. We were able to address the issue quickly. The software was disabled on the server side (i.e., Superfish's search engine) in January 2015.

Not Just Komodia

- PrivDog
  - “Your privacy is under attack!”
- Provides “private Web browsing”
  - Translation: replaces ads on webpages with other ads from “trusted sources”
- Re-signs certificates to MITM SSL connections
- Accepts self-signed certificates and turns them into trusted certificates
- Founded by the CEO of Comodo CA

Just Say No

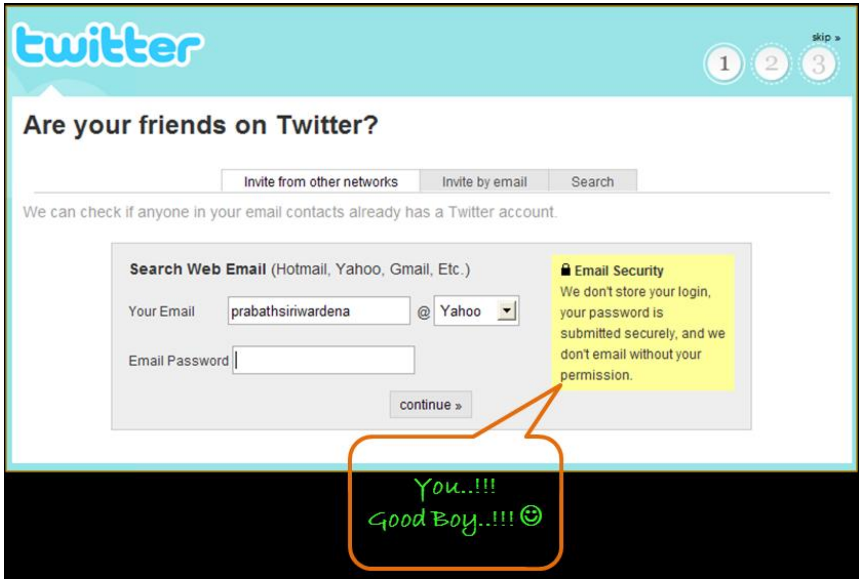


Credit: Adrienne Porter Felt (Google)

OAuth

The following slides shamelessly jacked from WS02

Before OAuth



Are your friends on Twitter?

Invite from other networks | Invite by email | Search

We can check if anyone in your email contacts already has a Twitter account.

Search Web Email (Hotmail, Yahoo, Gmail, Etc.)

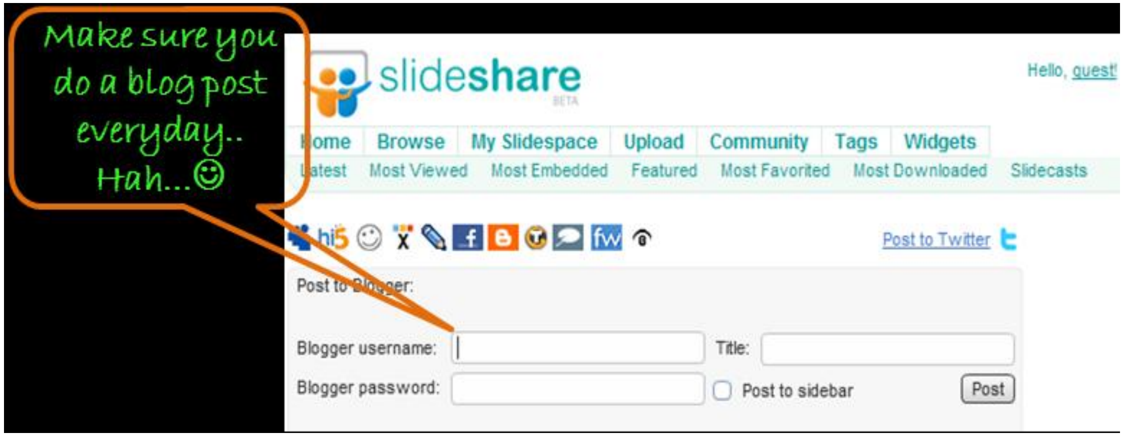
Your Email: prabathsiriwardena @ Yahoo

Email Password:

continue »

Email Security
We don't store your login, your password is submitted securely, and we don't email without your permission.

You..!!!
Good Boy..!!! ☺



Make sure you do a blog post everyday.. Hah...☺

slideshare BETA
Hello, guest!
ome Browse My Slidespace Upload Community Tags Widgets
atest Most Viewed Most Embedded Featured Most Favorited Most Downloaded Slidecasts
hi5
Post to Twitter

Post to Blogger:

Blogger username:

Title:

Blogger password:

□ Post to sidebar

Post

Issues (1)

- Third-party applications are required to store the resource owner’s credentials for future use, typically a password in cleartext
- Servers are required to support password authentication, despite the security weaknesses created by passwords

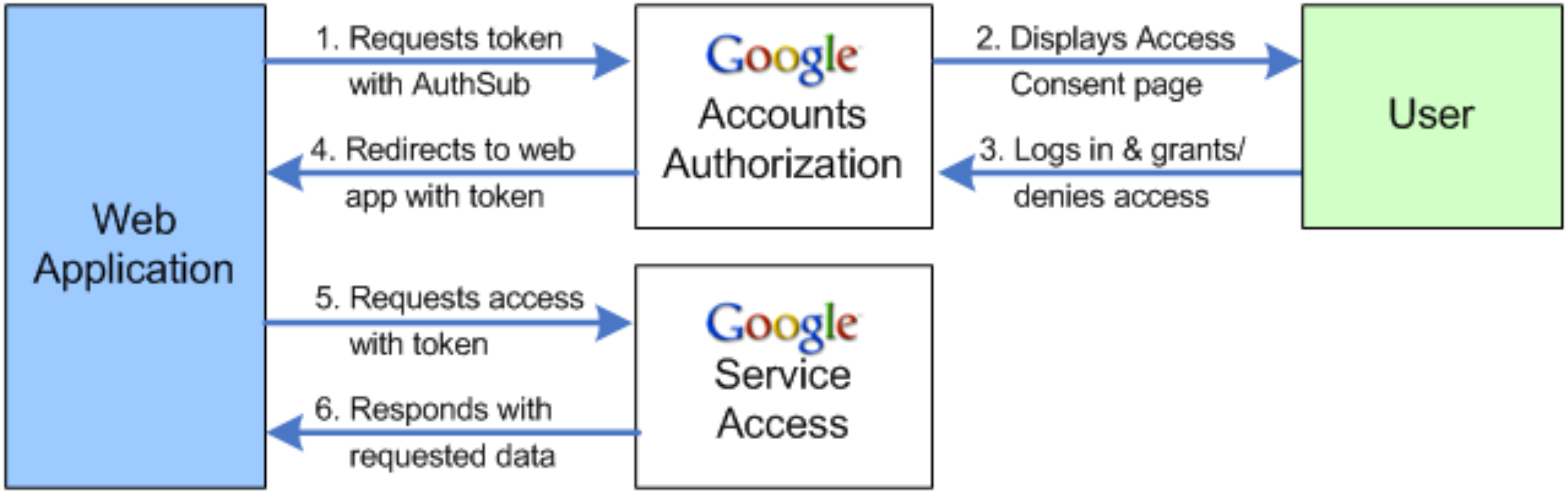
Issues (2)

- Third-party applications gain overly broad access to the resource owner’s protected resources, resource owners cannot restrict duration or access to a limited subset of resources
- Resource owners cannot revoke access to an individual third party without revoking access to all third parties, and must do so by changing their password
- Compromise of any third-party application results in compromise of the end-user’s password and all of the data protected by that password

Delegation

Before OAuth

Google calendar



Web Application

1. Requests token with AuthSub

Google Accounts Authorization

2. Displays Access Consent page

User

3. Logs in & grants/ denies access

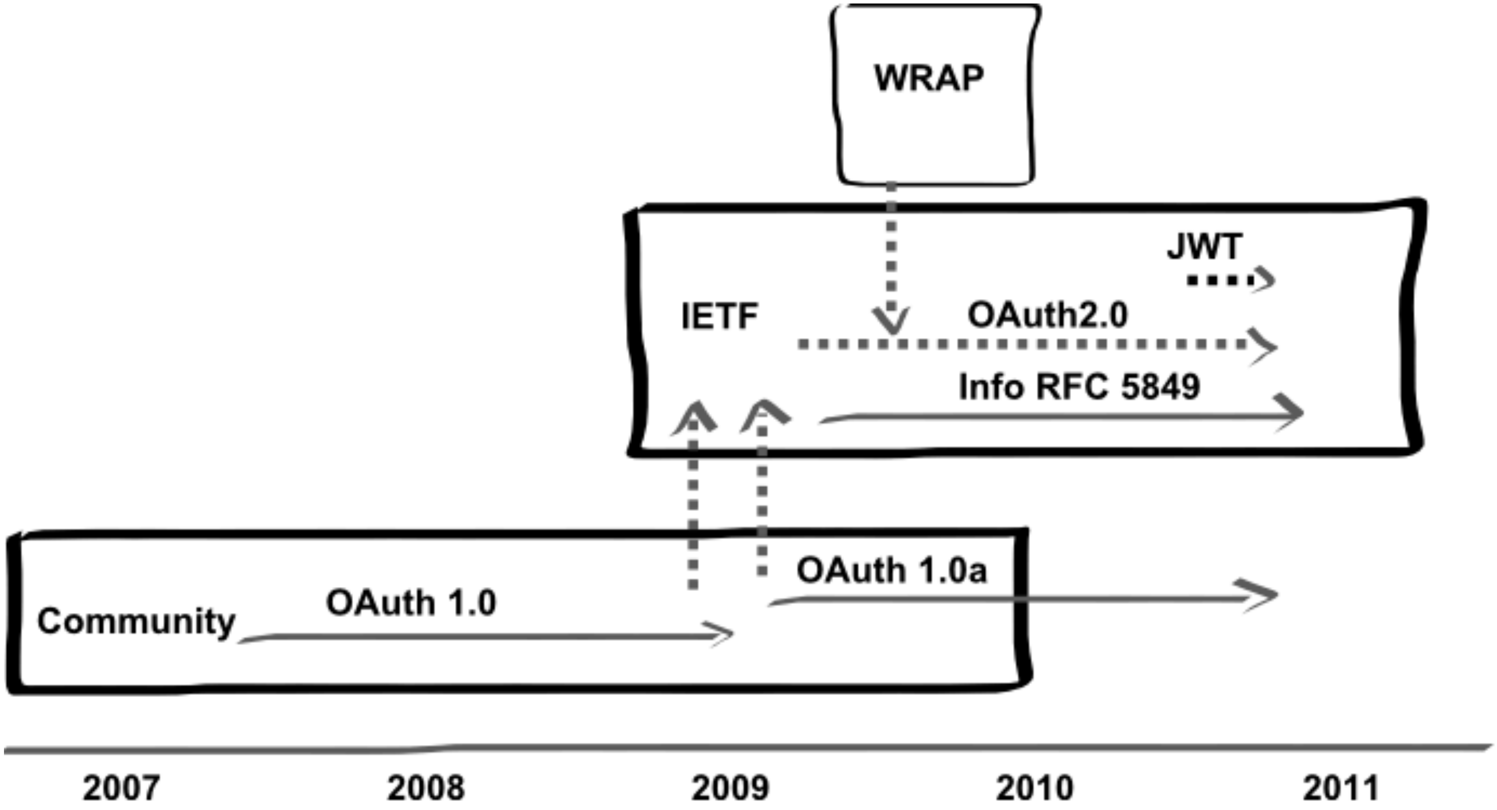
4. Redirects to web app with token

5. Requests access with token

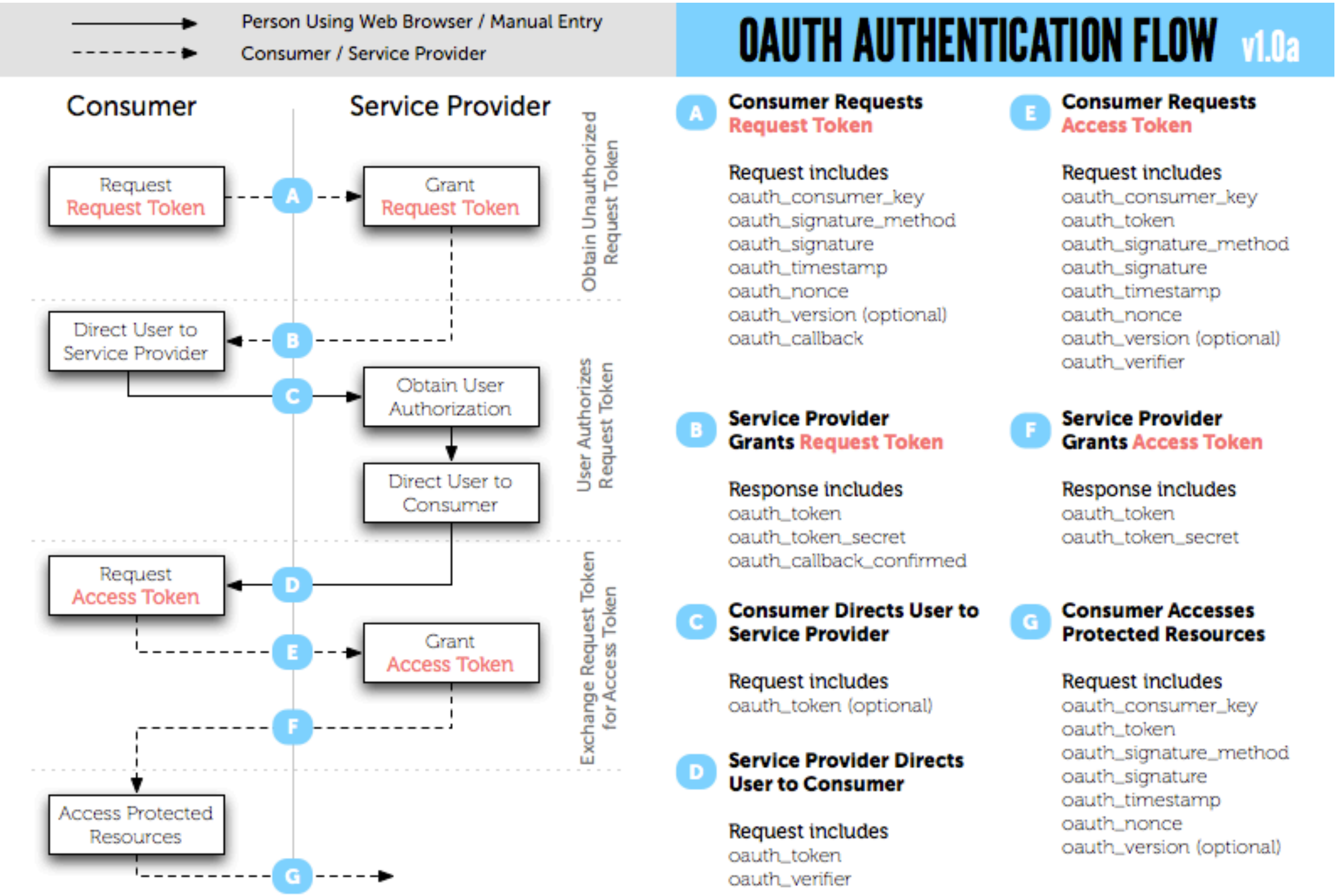
Google Service Access

6. Responds with requested data

OAuth Evolution



OAuth 1.0a



Person Using Web Browser / Manual Entry
Consumer / Service Provider
Consumer
Service Provider
Request Request Token
A
Grant Request Token
Obtain Unauthorized Request Token
Direct User to Service Provider
B
C
Obtain User Authorization
Direct User to Consumer
User Authorizes Request Token
Request Access Token
D
E
Grant Access Token
F
Exchange Request Token for Access Token
Access Protected Resources
G

OAUTH AUTHENTICATION FLOW v1.0a

A Consumer Requests Request Token

Request includes
oauth_consumer_key
oauth_signature_method
oauth_signature
oauth_timestamp
oauth_nonce
oauth_version (optional)
oauth_callback

B Service Provider Grants Request Token

Response includes
oauth_token
oauth_token_secret
oauth_callback_confirmed

C Consumer Directs User to Service Provider

Request includes
oauth_token (optional)

D Service Provider Directs User to Consumer

Request includes
oauth_token
oauth_verifier

E Consumer Requests Access Token

Request includes
oauth_consumer_key
oauth_token
oauth_signature_method
oauth_signature
oauth_timestamp
oauth_nonce
oauth_version (optional)
oauth_verifier

F Service Provider Grants Access Token

Response includes
oauth_token
oauth_token_secret

G Consumer Accesses Protected Resources

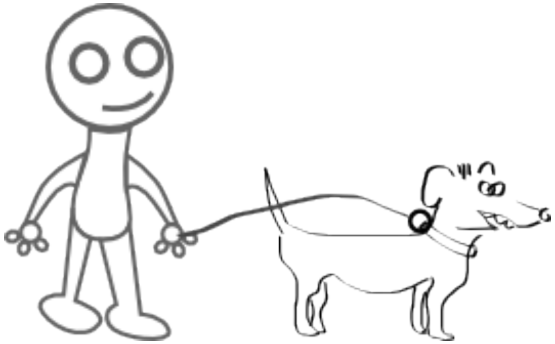
Request includes
oauth_consumer_key
oauth_token
oauth_signature_method
oauth_signature
oauth_timestamp
oauth_nonce
oauth_version (optional)

OAuth 1.0 Issues

- Complexity in validating and generating signatures
- No clear separation between Resource Server and Authorization Server
- Browser-based re-redirections

OAuth 2.0: Resource Owner

- An entity capable of granting access to a protected resource
- When the resource owner is a person, it is referred to as an end-user



OAuth 2.0: Resource Server

- The server hosting the protected resources, capable of accepting and responding to protected resource requests using access tokens

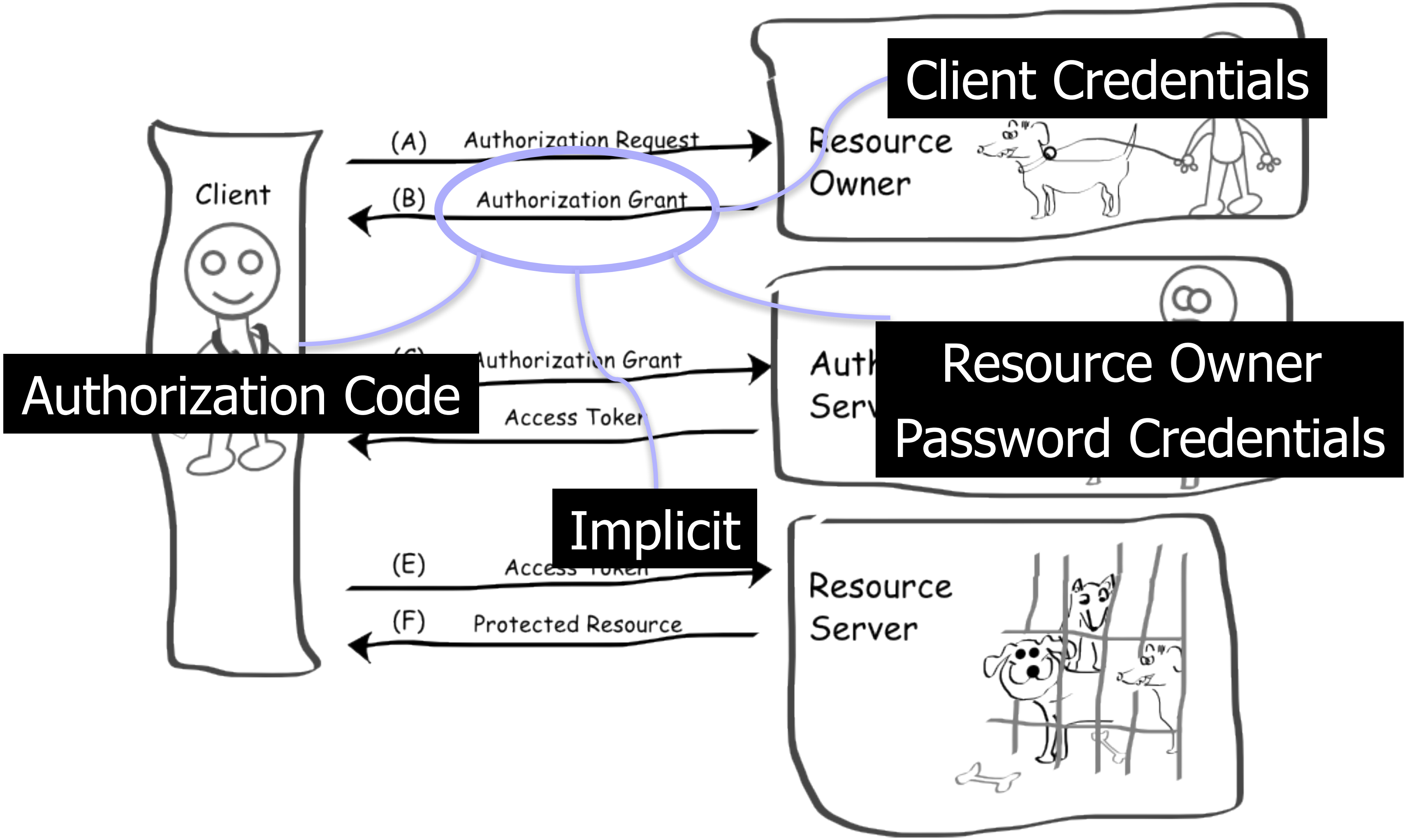
OAuth 2.0: Client

- An application making protected resource requests on behalf of the resource owner and with its authorization

OAuth 2.0: Authorization Server

- The server issuing access tokens to the client after successfully authenticating the resource owner and obtaining authorization

Authorization Grants in OAuth 2.0



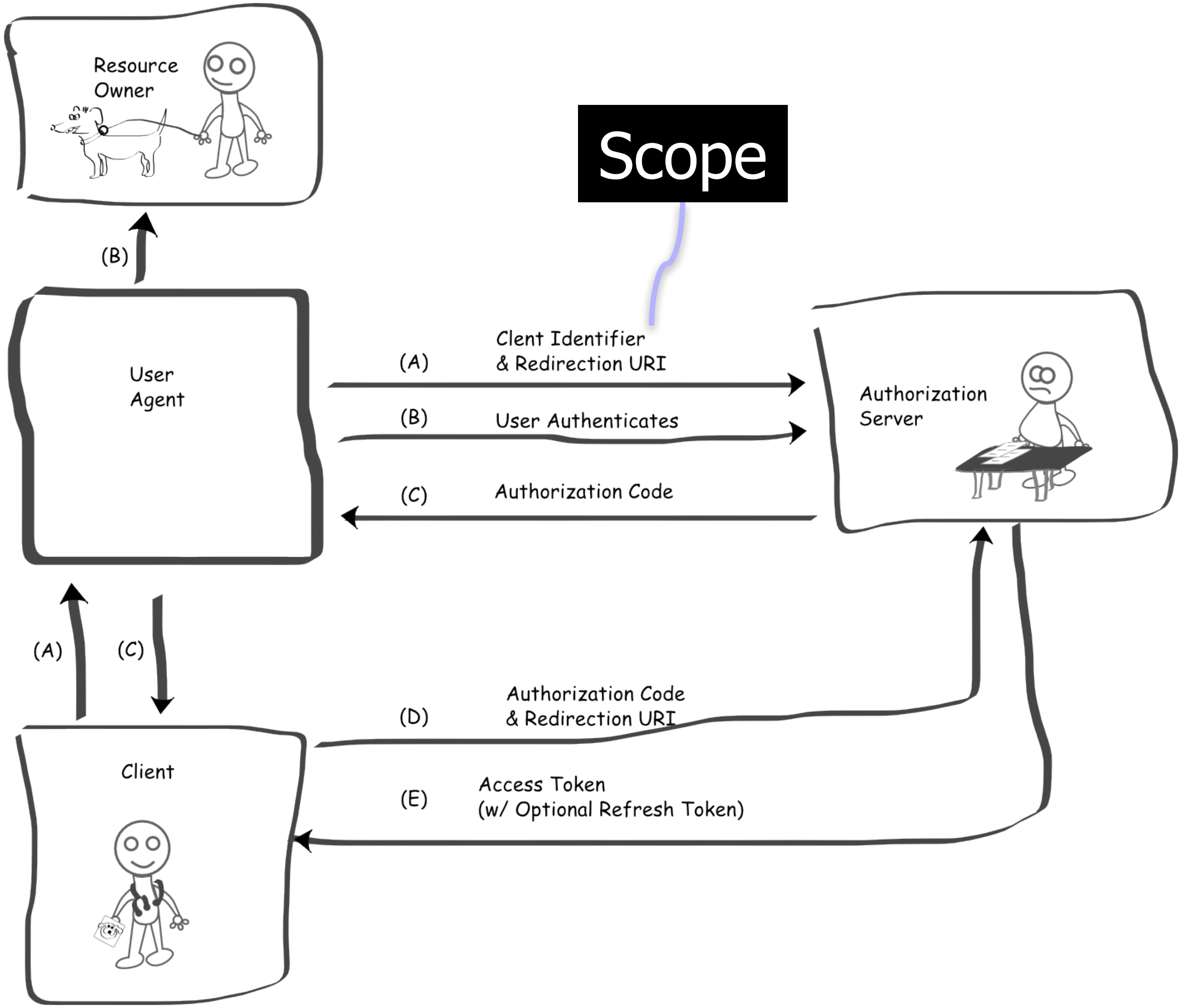
Client Credentials

Authorization Code

Implicit

Resource Owner Password Credentials

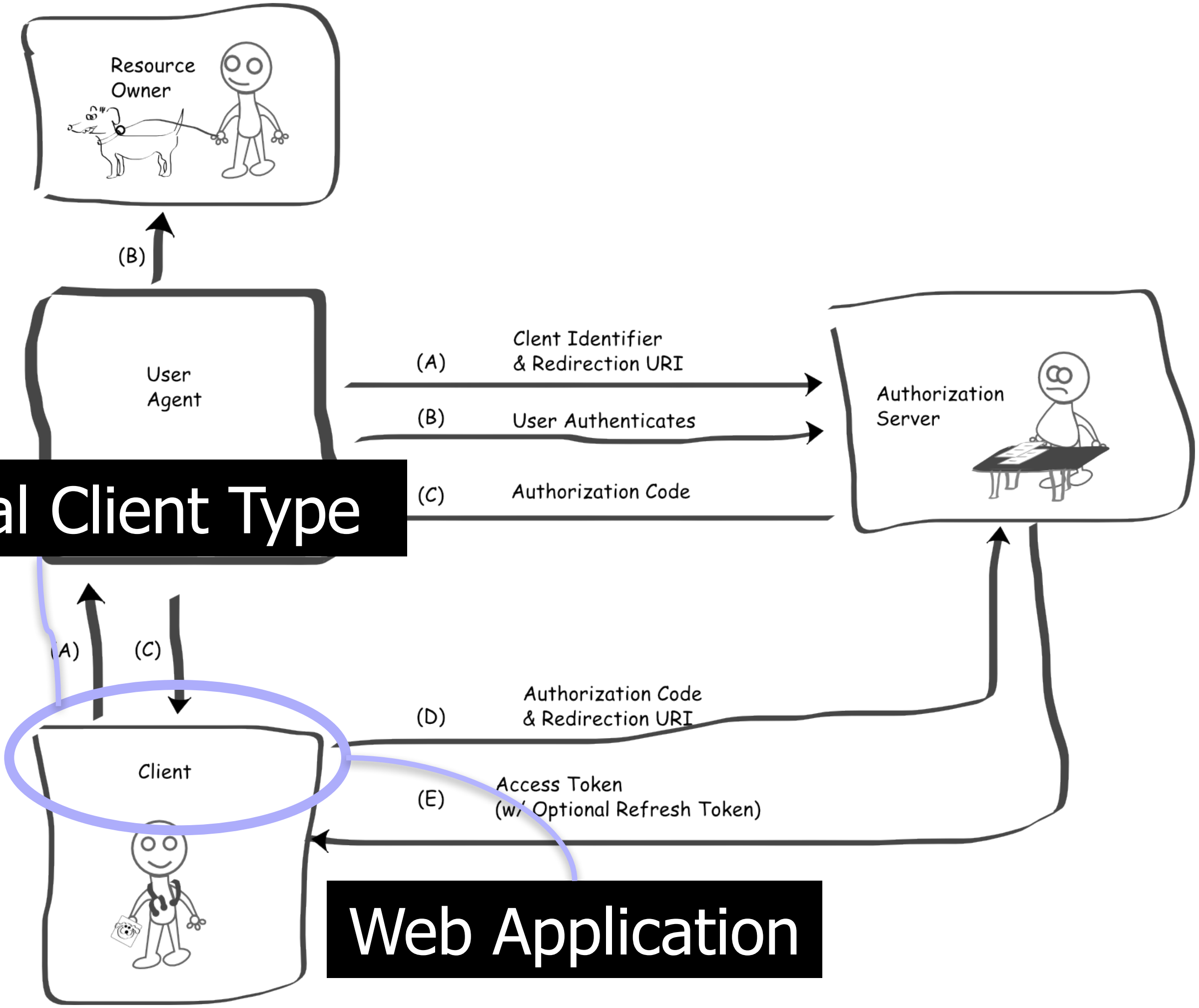
Authorization Code



Scope

- Defined by the Authorization Server
- Indicates what resource client wants access and which actions he wants to perform on that
- The value of the scope parameter is expressed as a list of space-delimited, case-sensitive strings
  - The strings are defined by the authorization server

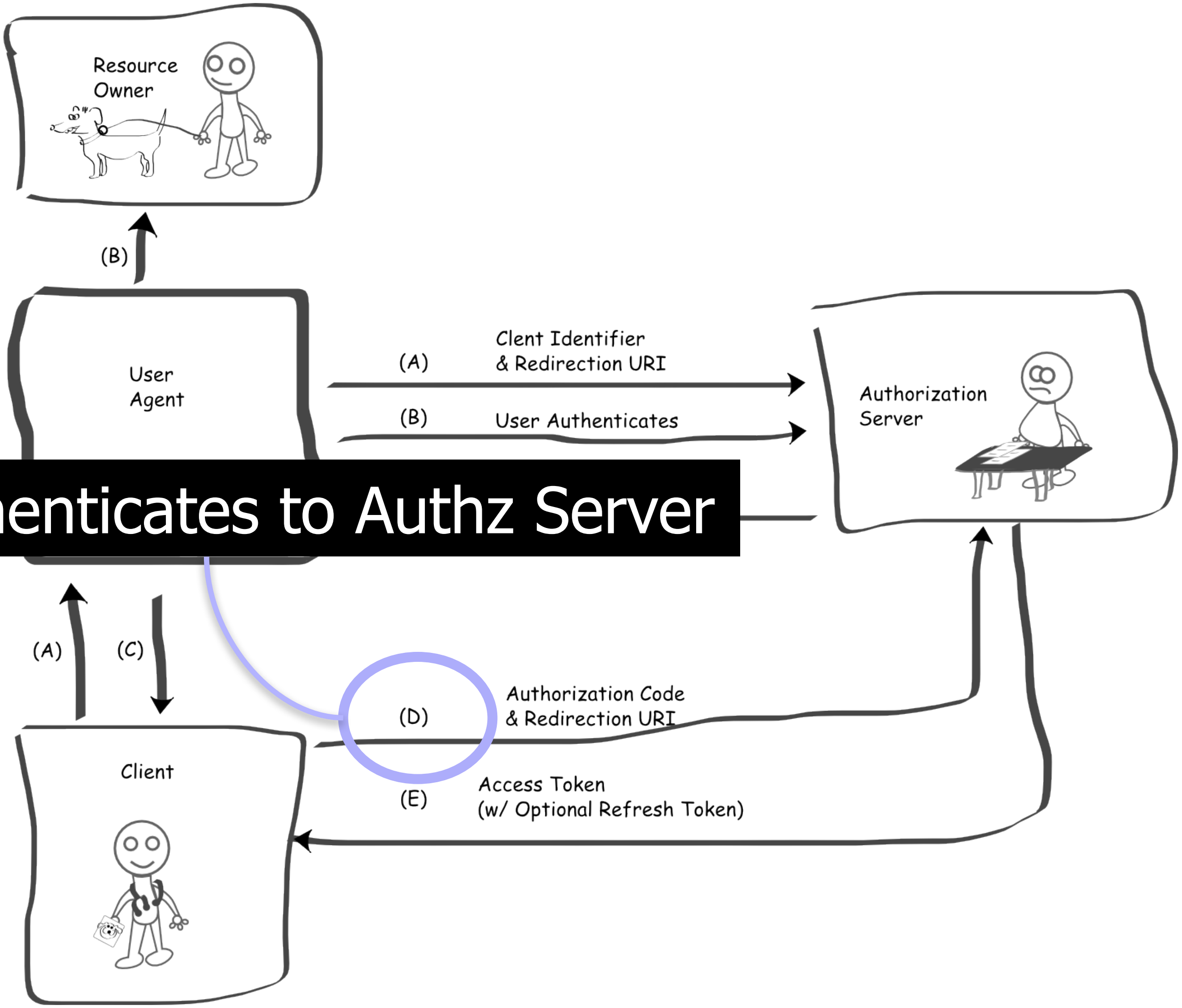
Authorization Code



Confidential Client Type

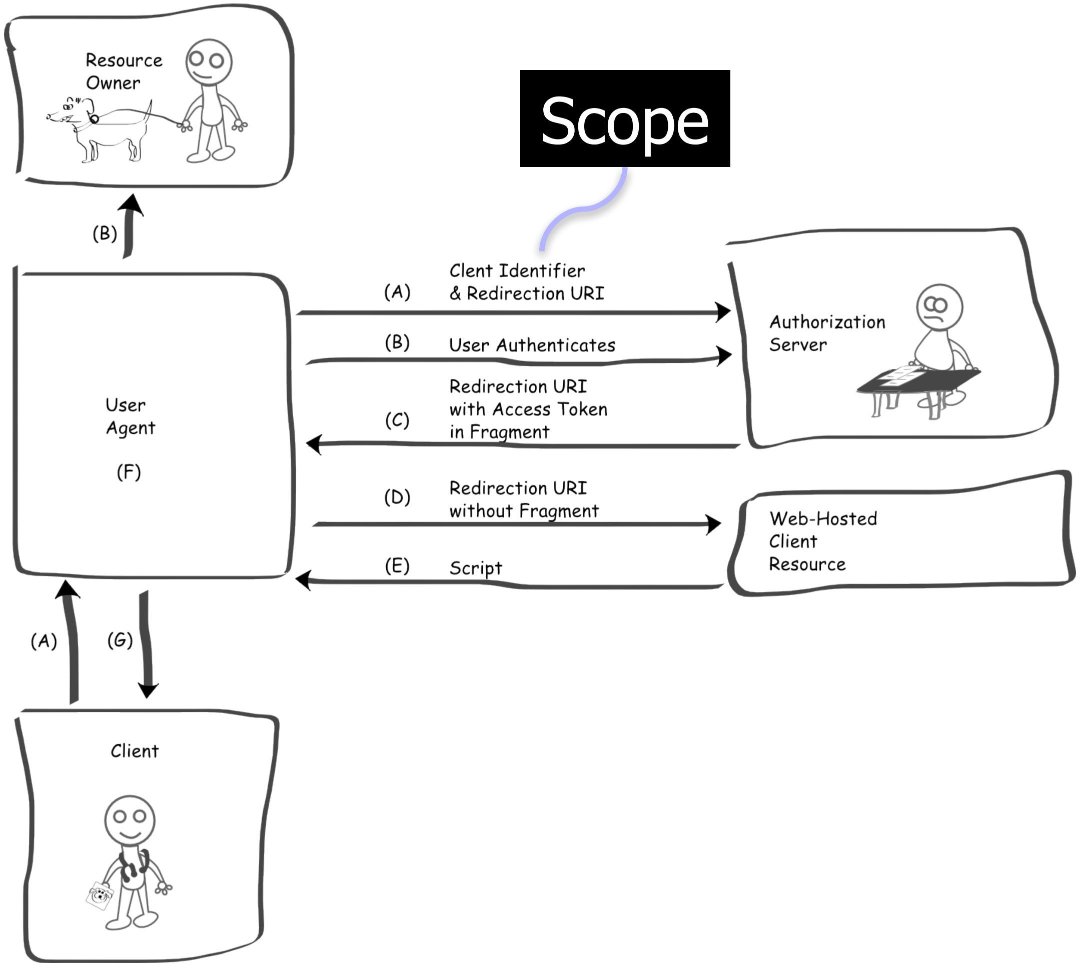
Web Application

Authorization Code



Client Authenticates to Authz Server

Implicit



Scope

Resource Owner

(B)

User Agent

(F)

(A) Clent Identifier & Redirection URI

(B) User Authenticates

(C) Redirection URI with Access Token in Fragment

(D) Redirection URI without Fragment

(E) Script

Authorization Server

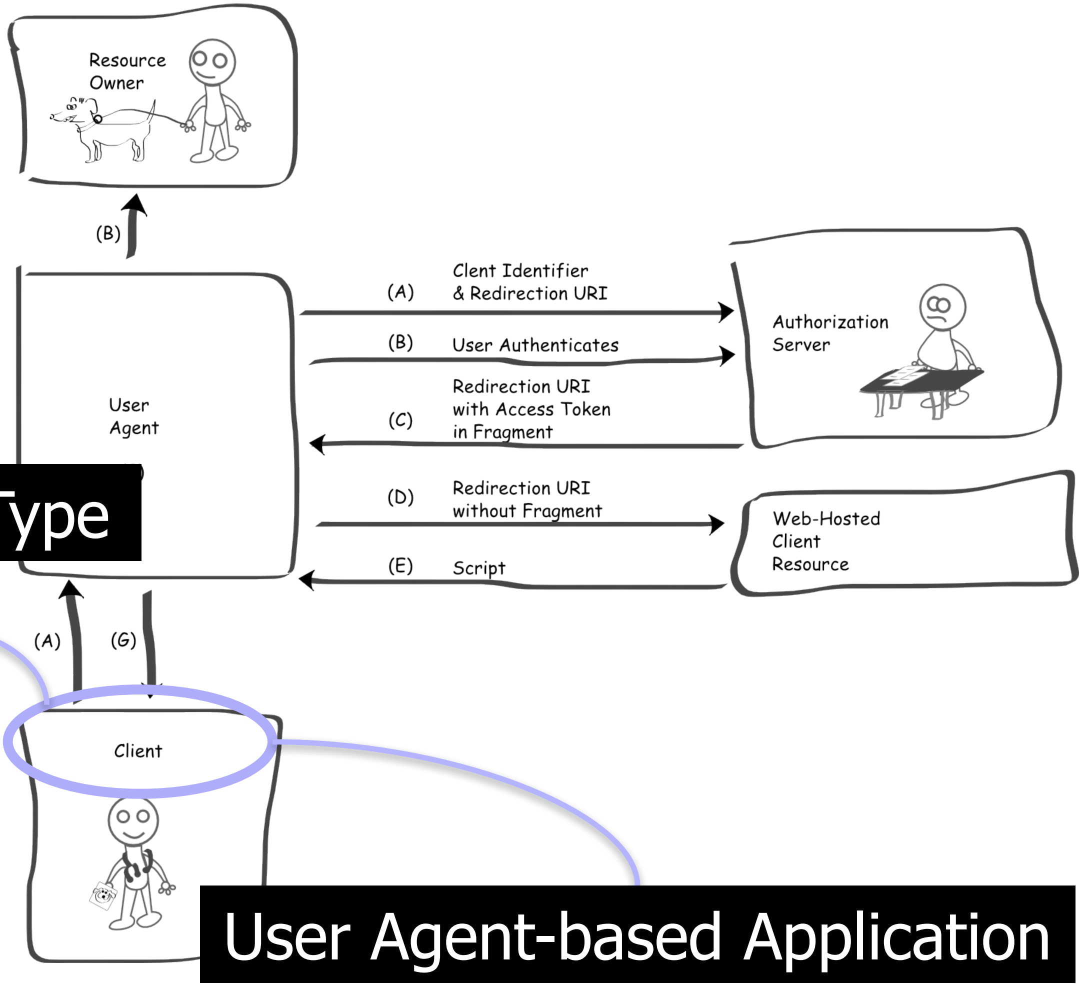
Web-Hosted Client Resource

(A)

(G)

Client

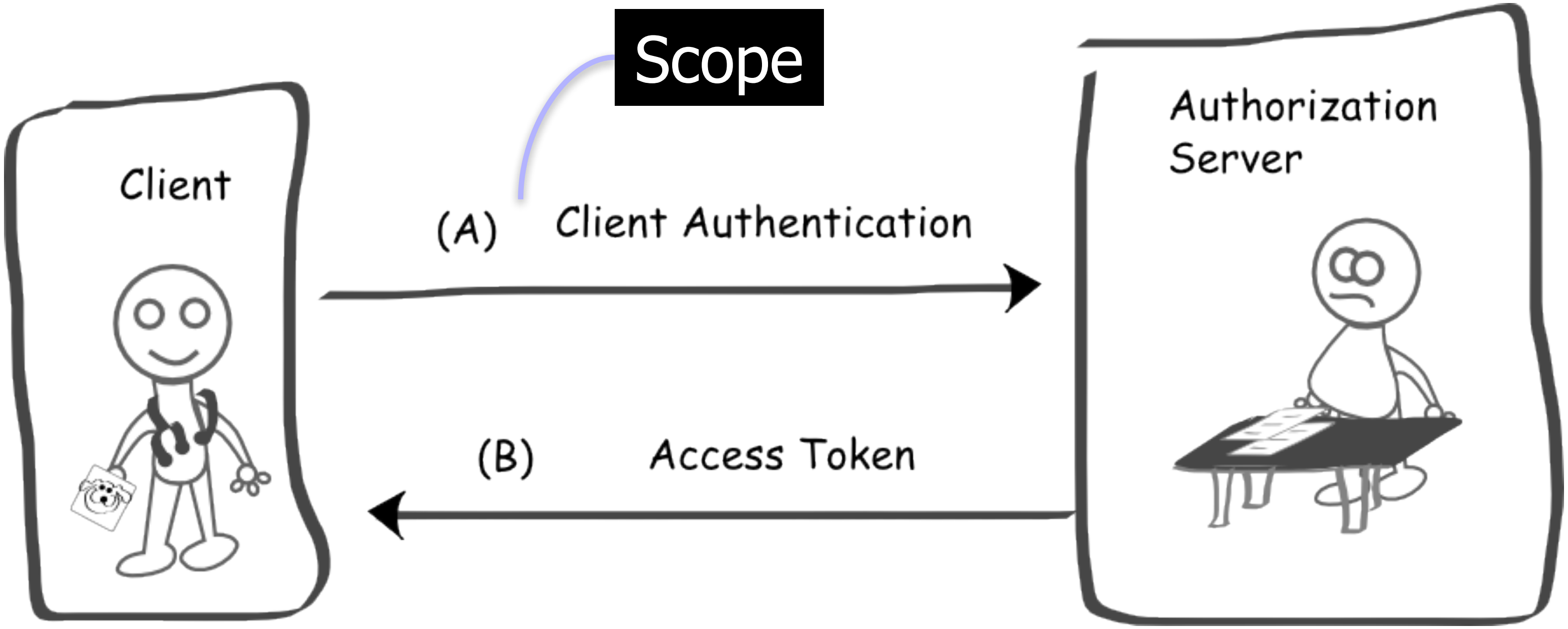
Implicit


Resource Owner
(B)
User Agent
(A)
Clent Identifier & Redirection URI
(B)
User Authenticates
Redirection URI with Access Token in Fragment
(C)
(D)
Redirection URI without Fragment
(E)
Script
Authorization Server
Web-Hosted Client Resource
(A)
(G)
Client

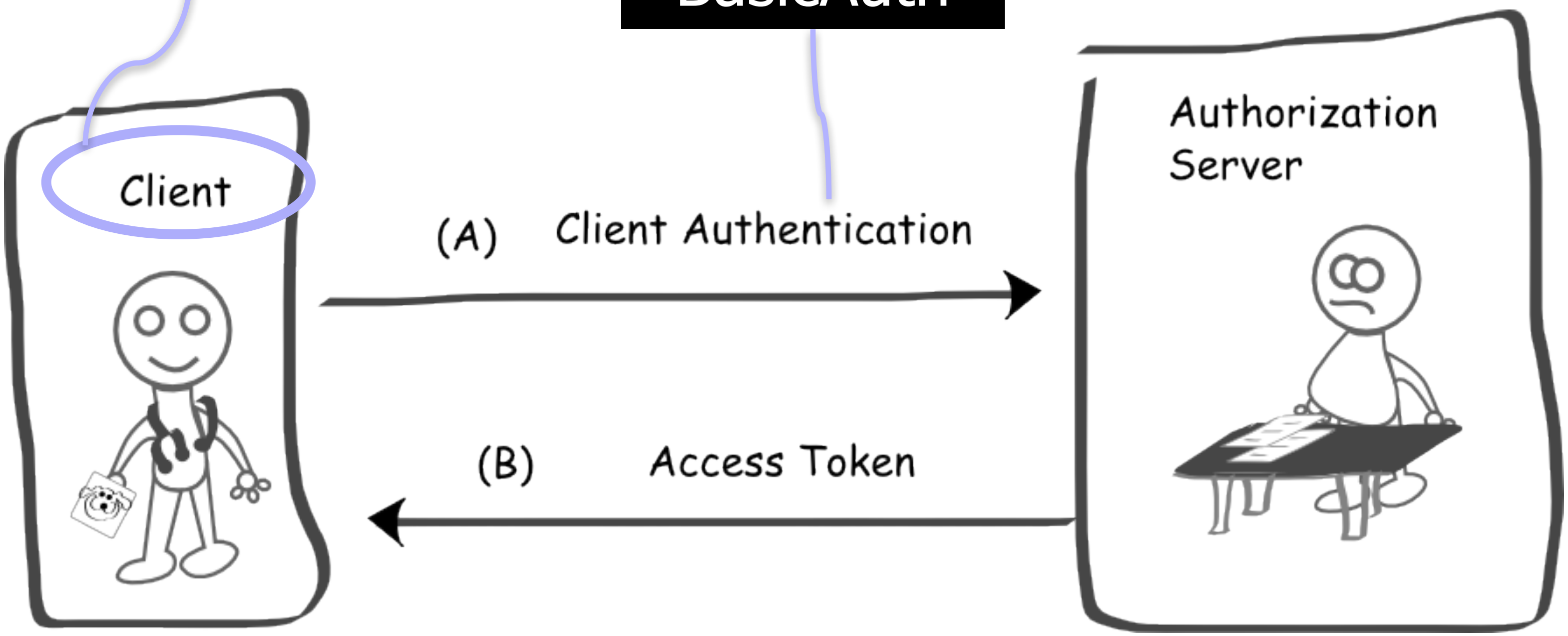
Public Client Type

User Agent-based Application

Client Credential



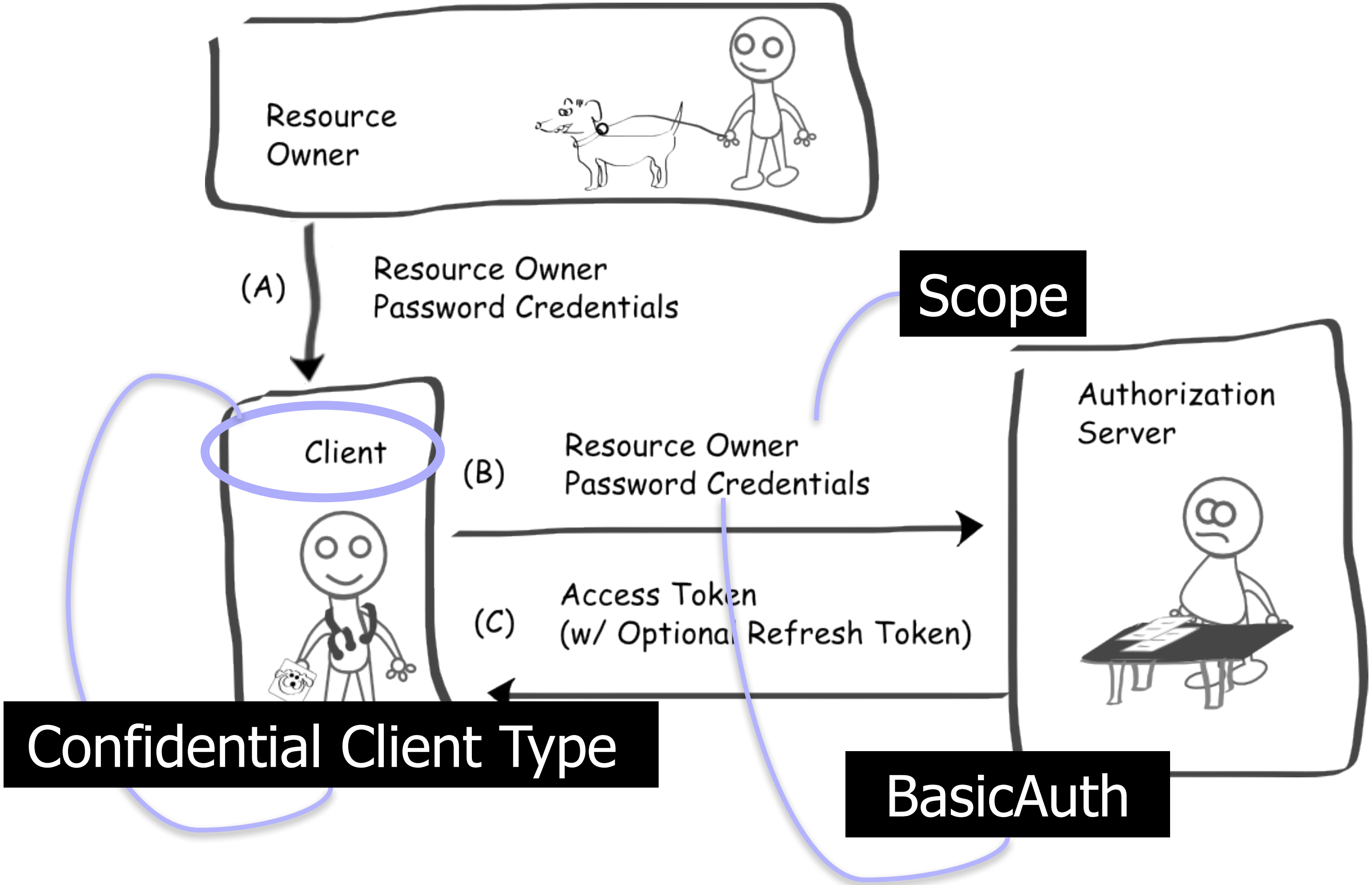
Client Credential



Confidential Client Type

BasicAuth

Resource Owner Pwd Credential



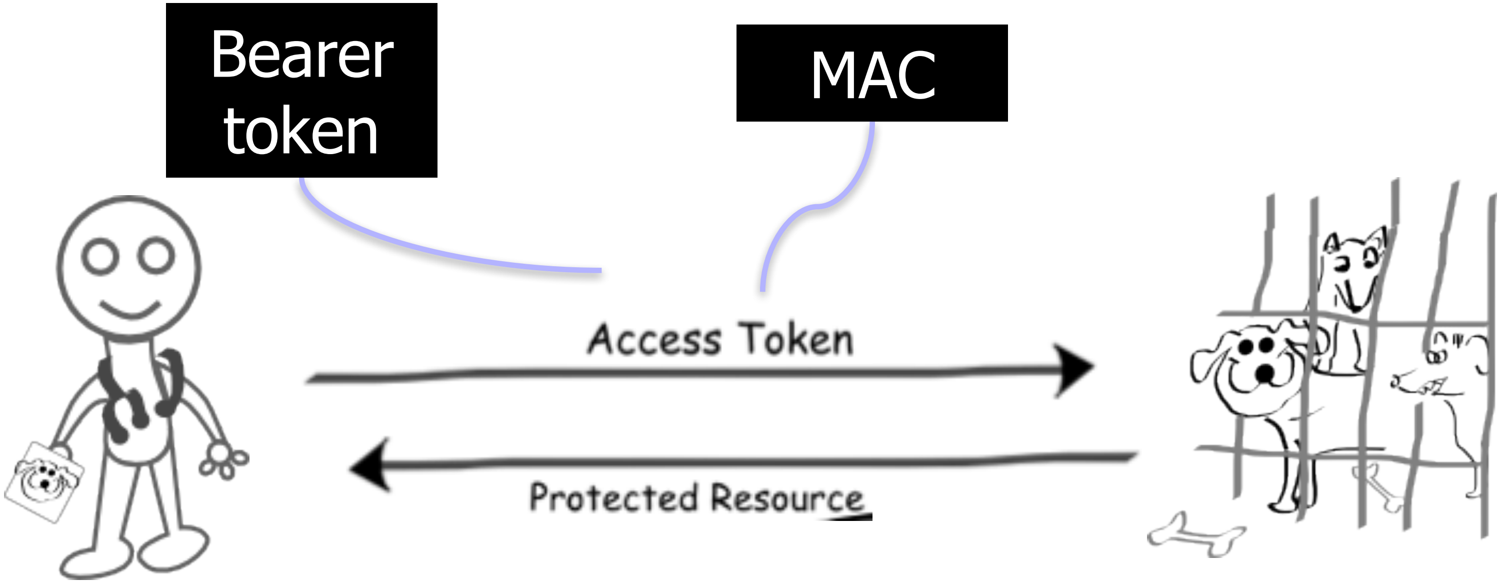
Resource Owner
(A)
Resource Owner Password Credentials
Client
(B)
Resource Owner Password Credentials
(C)
Access Token (w/ Optional Refresh Token)
Authorization Server

Scope

Confidential Client Type

BasicAuth

Runtime



Bearer token

MAC

Bearer Token and MAC

- Any party in possession of a bearer token (a "bearer") can use it to get access to the associated resources (without demonstrating possession of a cryptographic key)
- HTTP MAC access authentication scheme