

# Lecture 23

## Cookies and Sessions

ECE 422: Reliable and Secure Systems Design



Instructor: An Ran Chen  
Term: 2024 Winter

# Schedule for today

- Key concepts from last classes
- Cookies and sessions
- Ambient authority
- Signing cookies
- Cookie attributes
  - Domain attribute
  - Path attribute
  - Expires attribute

# Deadlock avoidance

To **avoid deadlock**, we can examine the processes and resources to guarantee there can never be a circular-wait condition.

- System has access to information in advance about what resources will be needed by which processes
- System only approves resource requests if the process can obtain all resources it needs in future requests

**However**, it is tough to avoid deadlock in practice

- Hard to determine all the needed resources in advance
- Good problem statement in theory, but difficult to apply in practice

# Resource Allocation Graph (RAG)

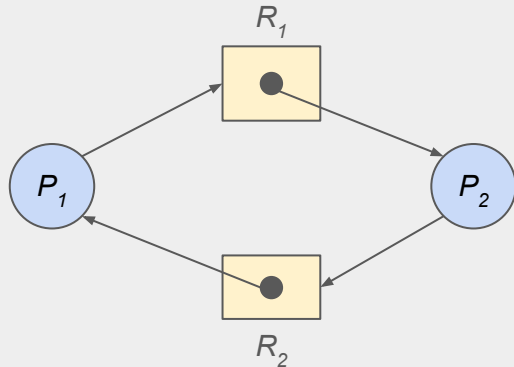
Resource Allocation Graph (RAG) represents the state of the system as graphs. The graph contains a set of vertices and edges to describe the state of the system:

- Vertices (nodes) can either represent a process or resource
  - Process:  $P = \{P_1, P_2, \dots, P_n\}$  is the set of processes in the system
  - Resource:  $R = \{R_1, R_2, \dots, R_m\}$  is the set of resources in the system
- Edges can either represent a request or assignment
  - Request edge (directed edge):  $P_i \rightarrow R_j$
  - Assignment edge (directed edge):  $R_j \rightarrow P_i$

# Basic facts

To determine if a resource allocation graph contains deadlock:

- If graph contains no cycle, then no deadlock
- If graph contains at least one cycle
  - If only one instance per resource, then deadlock
  - If several instances per resource, then possible deadlock



Example: processes in deadlock

- $P_1$  holds  $R_2$
- $P_1$  waits for  $R_1$
- $P_2$  holds  $R_1$
- $P_2$  waits for  $R_2$

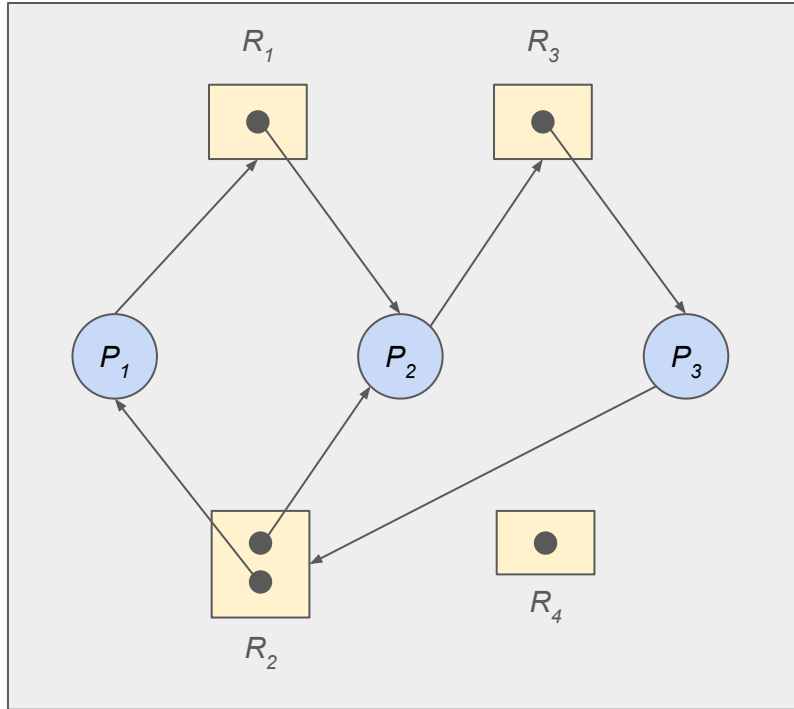
# Example 2: Resource Allocation Graph



Question: Is there a deadlock?

Step 1: Fill in the allocated resources

Step 2: Fill in the requested resources



	Allocated			Requested		
	R1	R2	R3	R1	R2	R3
P1						
P2						
P3						

Hint: allocated =  $R \rightarrow P$ ; requested =  $P \rightarrow R$

# Schedule for today

- Key concepts from last classes
- Cookies and sessions
- Ambient authority
- Signing cookies
- Cookie attributes
  - Domain attribute
  - Path attribute
  - Expires attribute

# Cookies

Cookies are small piece of data that a server sends to a user's web browser.

- Web browser may store the cookie and send it back to the same server with later requests

Goals of cookies:

- Personalization: provide experiences based on personal preferences
  - E.g., preferred language to automatically deliver the right content
- Session management: make it easier for users to access accounts
  - E.g., stored login information to avoid login every time
- Tracking: track and analyze how users interact with the website
  - E.g., user behavior to enhance performance





# Example to illustrate cookies

Example of how cookies work: <https://cookie-tracking.glitch.me/>

- First visit: Welcome to the website
- From next visit: Welcome back!

After clearing the cookies:

- Back to: Welcome to the website

**Welcome to the website!**



**Welcome back!**

# Session cookies

A session cookie (also known as non-persistent cookie) is created by the web server and used to store information about the user's browsing session.

- Session cookies are used by the server to implement sessions
- Deleted once the session ends (e.g., logged out)

Example of session cookies:

- Logins
- Shopping carts
- User tracking

# Creating session cookies

After receiving an HTTP request, a server can send **Set-Cookie** headers with the response:

- Use the Set-Cookie response header to send cookies from the server to the user agent.
- **Set-Cookie:** <cookie-name>=<cookie-value>
- E.g., **Set-Cookie:** theme=dark

# Creating session cookies

After receiving an HTTP request, a server can send **Set-Cookie** headers with the response:

- Use the Set-Cookie response header to send cookies from the server to the user agent.
- **Set-Cookie:** <cookie-name>=<cookie-value>
- E.g., **Set-Cookie:** theme=dark

Then, with every subsequent request to the server, a client can send **Cookie** headers with a request:

- Use the Cookie request header to send cookies back to the server.
- **Cookie:** <cookie-name>=<cookie-value>
- E.g., **Cookie:** theme=dark

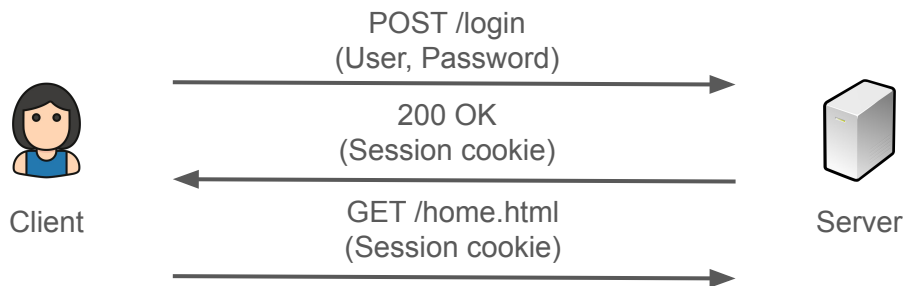
# Login example with session cookies

Alice needs to login into her account:

- Alice sends her credentials to the server
- Server verifies the identity and authentication of Alice
- If credentials match, server returns 200 OK and a session cookie

For all future requests:

- Alice uses her session cookie to authenticate



# Schedule for today

- Key concepts from last classes
- Cookies and sessions
- **Ambient authority**
- Signing cookies
- Cookie attributes
  - Domain attribute
  - Path attribute
  - Expires attribute

# Ambient authority

Ambient authority is an access control based on global and persistent properties of the requester.

- Involve implicit trust and privileges to a user

In the context of web application security, the authority is automatically granted to a user based on their session state

For example:

- User logs into a web application
- Server sends the session ID back to the user
- Ambient authority: web application implicitly authenticates and authorizes all the actions performed by the user based on the **session ID**

# Ambient authority

There are four types of ambient authority on the web:

- **(Session) Cookies**: most common, most versatile method
- **IP checking**: used at some of the UoA printing services
  - Problem: Possible to change the IP address by having a server on campus
- **Client certificates**: rarely used
  - Verify the owner's identity with more than just a signature
  - Problem: Require a client application to submit a client certificate for authentication
- **Basic (HTTP) authentication**: rarely used
  - Problem: Authentication headers can be seen, which makes it easier to capture user credentials
  - In 2022, Microsoft disabled Basic authentication on Outlook

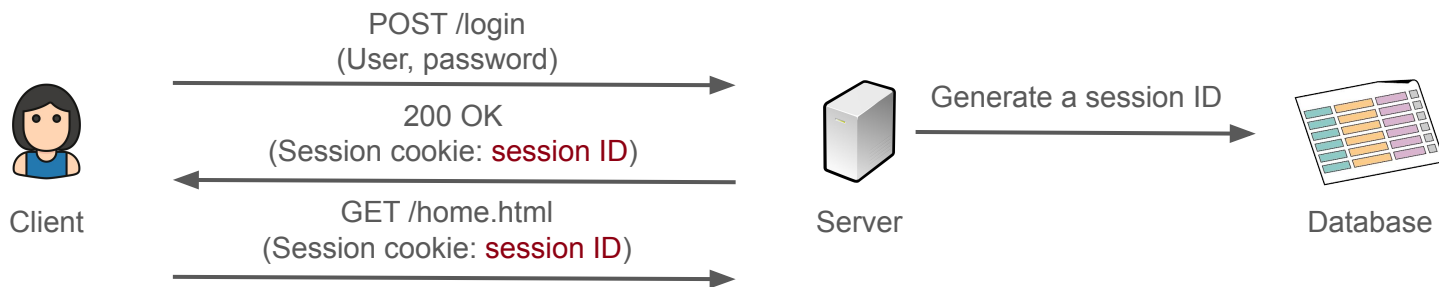


# Session ID

A session ID (also known as session token) is a unique identifier that a web server assigns to a user for the duration of the user's current session.

- Randomly generated token
- Analogy: Coat tickets at a restaurant

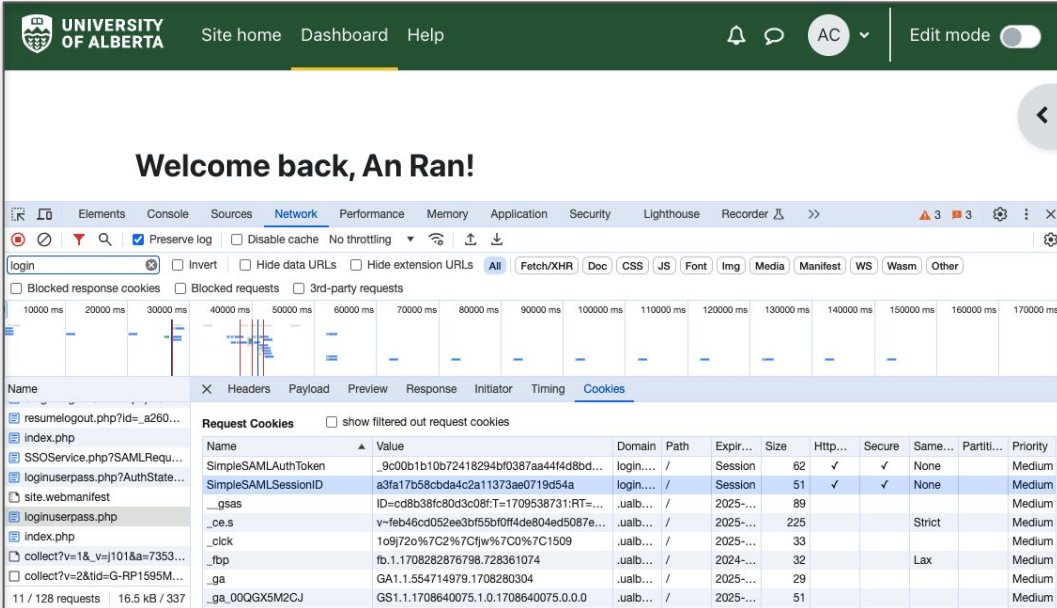
After the server verifies the credentials, it will generate a login event and returns a session ID in the form of a cookie.



# Example of session ID: eClass

You can inspect the network traffic from eClass login to see an example:

- Enable Preserve log checkbox in Network tab
- Login into eClass
- loginuserpass.php will appear in the log
  - Switch to Cookies tab
  - SimpleSAMLSessionID is the session ID



The screenshot shows the University of Alberta eClass login page. The Chrome DevTools Network tab is open, displaying a list of requests. The 'loginuserpass.php' request is selected, and the 'Cookies' sub-tab is active. The table below shows the cookies returned in the response.

Name	Value	Domain	Path	Expir...	Size	Http...	Secure	Same...	Partiti...	Priority
SimpleSAMLAuthToken	_9c00b1b10b72418294bf0387aa44d8bd...	login...	/	Session	62	✓	✓	None		Medium
SimpleSAMLSessionID	a3fa17b58cbda4c2a11373ae0719d54a	login...	/	Session	51	✓	✓	None		Medium
__gsas	ID=cd8b38fc80d3c08f:T=1709538731:RT=...	.ualb...	/	2025-...	89					Medium
__ce.s	v-feb46cd052ee3bf55bf0ff4de804ed5087e...	.ualb...	/	2025-...	225			Strict		Medium
__clk	109j72o%7C2%7Cfjw%7C0%7C1509	.ualb...	/	2025-...	33					Medium
__fbp	fb.1.1708282876798.728361074	.ualb...	/	2024-...	32			Lax		Medium
__ga	GA1.1.554714979.1708280304	.ualb...	/	2025-...	29					Medium
__ga_00OGX5M2CJ	GS1.1.1708640075.1.0.1708640075.0.0.0	.ualb...	/	2025-...	51					Medium

# Example of database with login events

The screenshot displays a web-based interface for managing sessions. At the top, there's a 'Sessions' tab and a search bar with the instruction 'Use the search tool to find Sessions.' Below this, a search panel contains various filters: Application ID, User ID, Username, Session ID, Internal Session ID, Authentication Status, Client Type, IP Address, Device ID, Processing Status, Alert Level, and Login Time. The 'Search' button is highlighted. Below the search panel, the 'Search Results' section shows a table of results. The table has columns for Session ID, Internal Session ID, Alerts, Application ID, Username, Device ID, IP Address, Location, and Auth Status. The results are filtered by 'High Alerts: (1)' and 'Medium Alerts: (1)'. The table shows 15 rows of data, including session IDs, internal session IDs, application IDs, usernames, device IDs, IP addresses, locations, and authentication statuses.

Session ID	Internal Session ID	Alerts	Application ID	Username	Device ID	IP Address	Location	Auth Status
3760	26_7ea78ffc3818fce	High Alerts: (1)	bharosaUIOGrp	gdfig	3201	141.144.80.177	denmark, kopenh	Success
3759	21_bf6d8d53789551e		bharosaUIOGrp	kama1216	3201	141.144.80.177	denmark, kopenh	Wrong
3758	16_904940b56c76f8e		bharosaUIOGrp	bodurai	3201	141.144.80.177	denmark, kopenh	Success
3757	12_34922f8a765f3d1		bharosaUIOGrp	bodurai	3201	141.144.80.177	denmark, kopenh	Wrong
3756	8_0603273ecbd5389e		bharosaUIOGrp	bodurai345	3201	141.144.80.177	denmark, kopenh	Wrong
3755	3_1f0679f07658b63e		bharosaUIOGrp	bodurai345	3201	141.144.80.177	denmark, kopenh	Pending
3754	43_4806da7b8f14af6		bharosaUIOGrp	dg	3201	141.144.80.177	denmark, kopenh	Success
3753	36_f2fa9d36922cc6d	Medium Alerts: (1)	bharosaUIOGrp	oam_two	3203	10.143.234.152	Private, Private, IBlocke	
3752	32_e0a40bf79c258f8		bharosaUIOGrp	oam_test	3203	10.143.234.152	Private, Private, IBlocke	
3751	28_48ad2f18822eb1e		bharosaUIOGrp	oam_two	3203	10.143.234.152	Private, Private, IBlocke	
3750	22_16f17f069b8378c		bharosaUIOGrp	surai	3201	141.144.80.177	denmark, kopenh	Success

# Login example in details

An actual example of session cookie:

s%3Al3ozSdvQ83TtC5RvJ.CibaQoHtaY0H3QOB1kqR8H2A

Signed/unsigned cookie

Session ID

Signature

- s%3A states it is a signed cookie
- l3ozSdvQ83TtC5RvJ indicates the session ID
- CibaQoHtaY0H3QOB1kqR8H2A shows the signature

Signing is necessary to confirm that the cookie originated from the server.

# Digital signature

Digital signatures verify the authenticity

- Detect the identity of the sender/signer/requester

Digital signatures check the integrity

- Verify that the message was not changed

Digital signatures ensure non-repudiation

- Verify that the signature is not fake

# Session fixation attack

A third party is able to determine a user's session identifier (i.e., by reading it or setting it), and therefore interact with the server as that user.

- Stealing cookies is one way to do this

A more advanced technique:

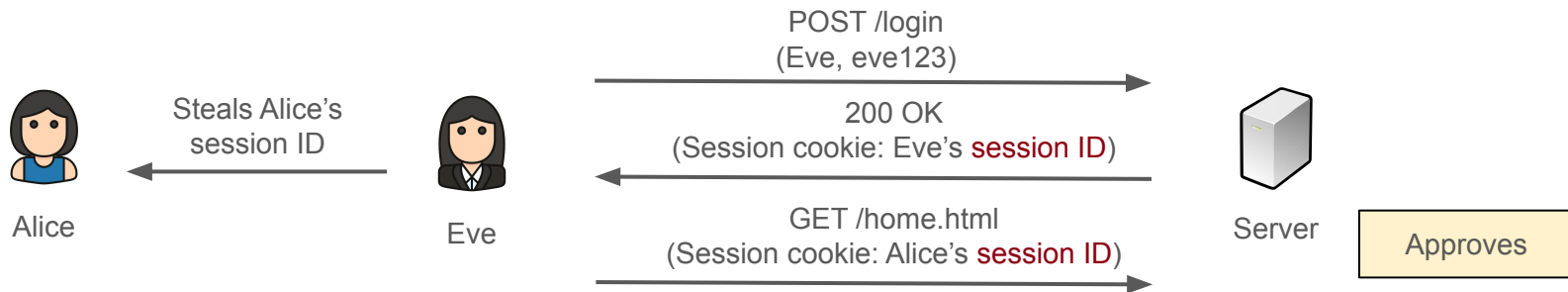
- When the user visits a page on the parent domain (or another subdomain), the application may share the existing value sent in the user's cookie
- This could allow an attacker to hijack a session after the user logs in

# Without signing cookies

Without signature, another client can access someone else's session. For example:

- Eve can copy over Alice's session ID to her browser
- Server validates the session ID (which is valid) and approves the request
- **Confidentiality** of the system is broken!

Signature ensures the integrity of the cookie



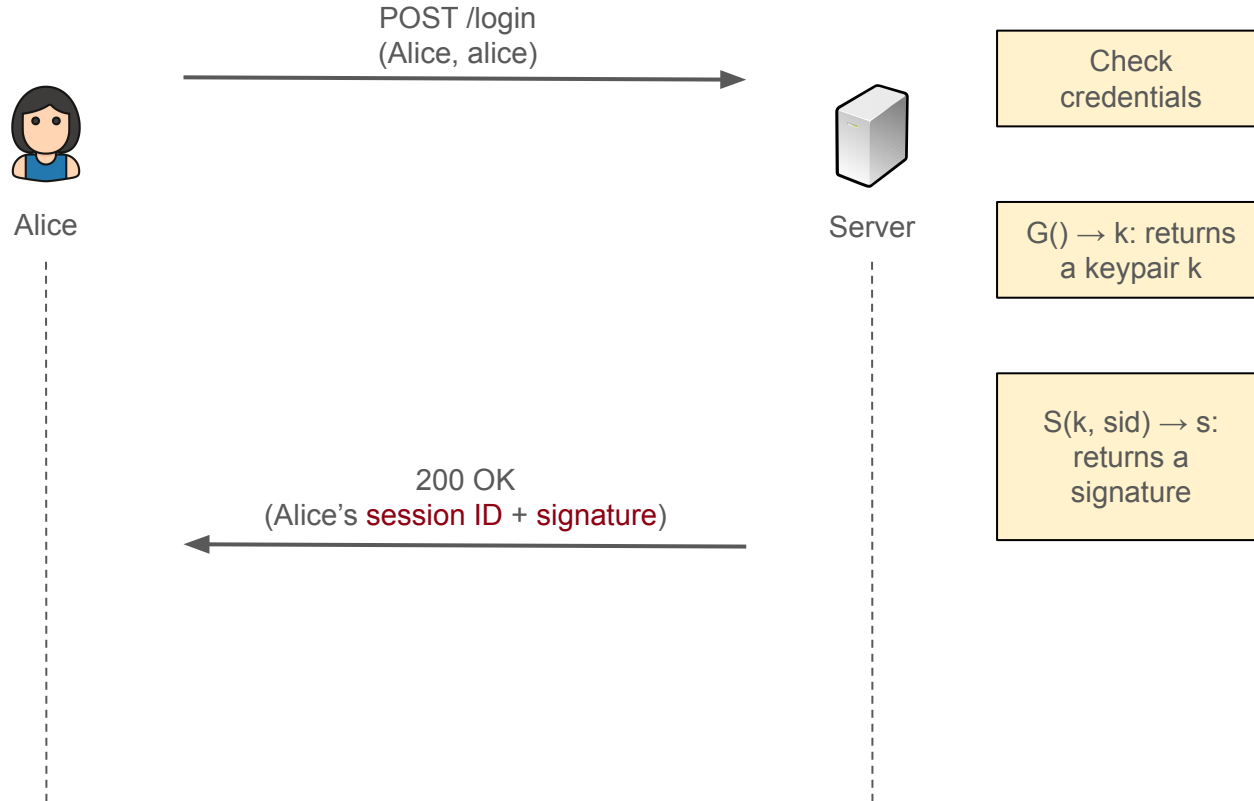
# Signing cookies

A signature scheme consists of a triple of algorithms (G, S, V):

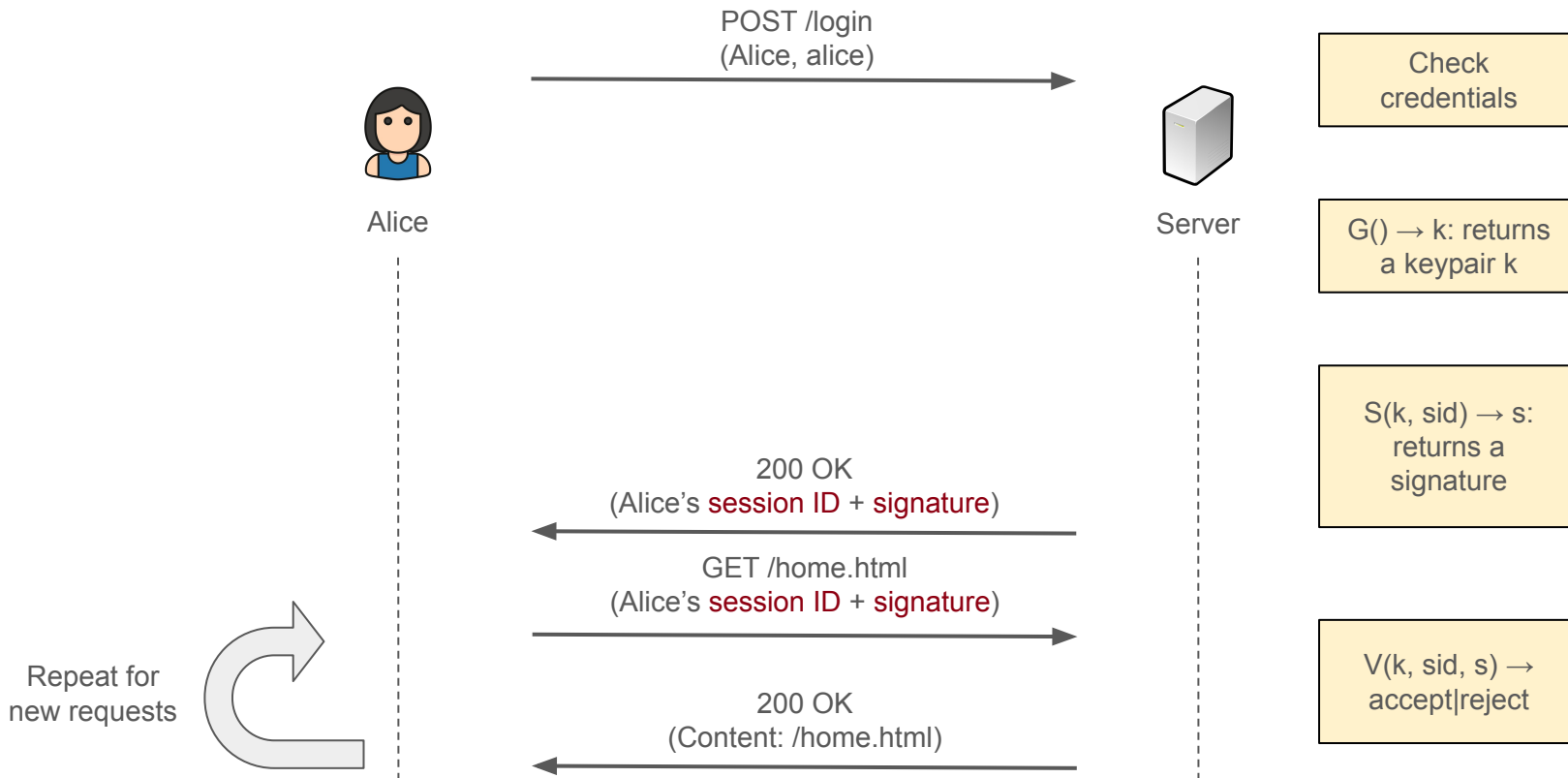
- G = key generation algorithm
  - Generate a public key for verification
  - Generate a private key for signing
  - $G() \rightarrow k$ : returns a keypair  $k$
- S = signing algorithm
  - Output a signature
  - $S(k, \text{sid}) \rightarrow s$ : returns a signature for keypair  $k$  and session ID  $\text{sid}$
- V = verification algorithm
  - Output a value stating the validity of the signature (e.g., accept/reject)
  - $V(k, \text{sid}, s) \rightarrow \text{accept|reject}$ : checks validity of signature for given key pair  $k$  and session ID  $\text{sid}$



# Signing cookies



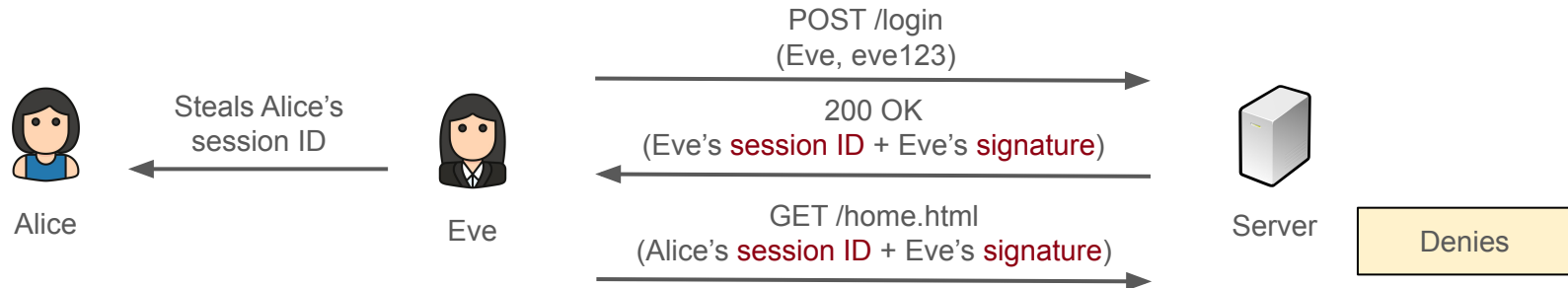
# Signing cookies



# Signing cookies

By signing the cookie, we make sure that the cookie has not been modified.

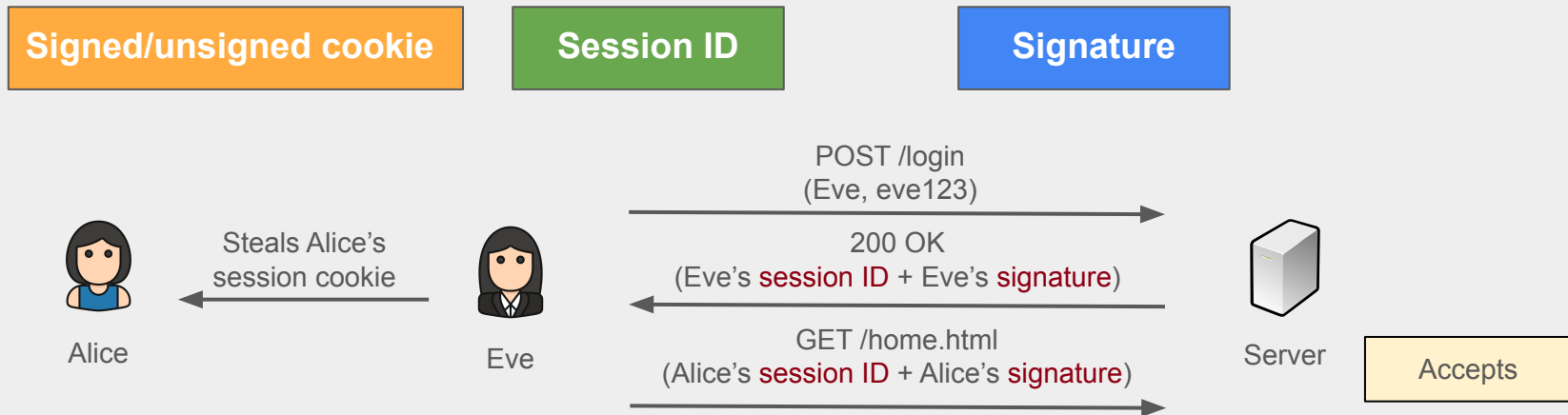
- Eve can copy Alice's session ID over to her browser
- Server validates the signature with the session ID (which is invalid)
- Server denies the request as the signature does not match the session ID
- **Confidentiality** of the system is satisfied



# Another attempt to break the system

What if the whole session cookie was stolen?

s%3Al3ozSdvQ83TtC5RvJ.CibaQoHtaY0H3QOB1kqR8H2A



**Confidentiality** of the system is broken again!

# Password reset on leaked session cookie



Can we fix this use case by resetting the password?

# Password reset on leaked session cookie

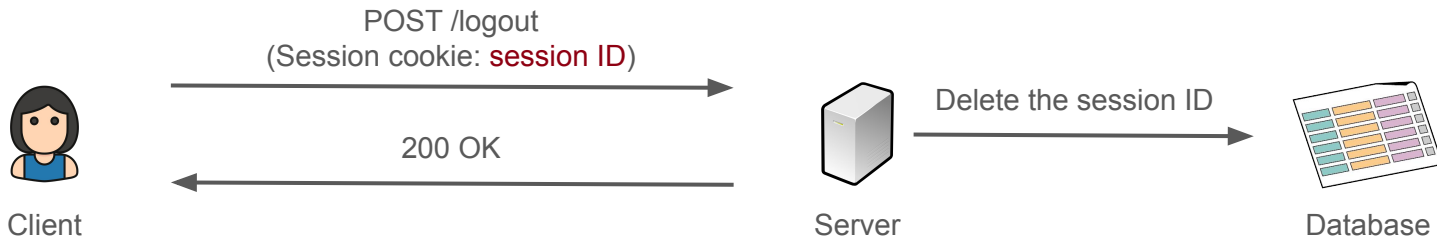


Can we fix this use case by resetting the password?

- Password may be reset, but the session ID and signature remain valid for authentication

Solution? We must destroy the session in a meaningful way

- Delete the session ID when the user resets the password, or regenerate and resend cookie when the user authenticate (even if already exist one)
- Analogy: Coat tickets at a restaurant



# Cookie attributes

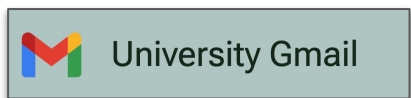
Cookie attributes are what protect the cookies from malicious users:

- Domain attribute: allows the cookie to be set to a broader domain
- Path attribute: scopes the cookie to a path prefix
- Expires attribute: specifies an expiration date
  - Expires date and time are relative to client the cookie is being set on, not the server
  - Defining lifetime of a cookie is necessary to avoid session fixation attacks

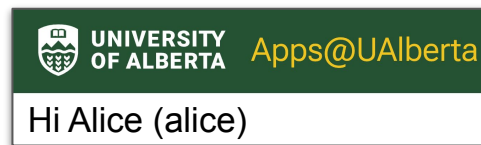
# Domain attribute

**Domain attribute** allows the cookies to be scoped to a broader domain.

- E.g., **Gmail Apps@UAlberta** could set a cookie for **Apps@UAlberta**
- Try it yourself in Incognito mode:
  - Login into your Gmail: [apps.ualberta.ca/appslink/auth/feature/gmail](https://apps.ualberta.ca/appslink/auth/feature/gmail)
  - Open [apps.ualberta.ca/](https://apps.ualberta.ca/), you should be logged in



Login into Gmail in  
Apps@UAlberta



Also logged in at  
Apps@UAlberta

If the domain attribute is set too **loosely**, then the server be may vulnerable to **session fixation attack** (e.g., allowing a third party to access the session id).



# Path attribute

**Path attribute** scopes the cookie to a path prefix, which works in conjunction with the domain attribute to a particular request path prefix.

- E.g., cookies in **path=/docs** will match **path=/docs/admin**
- Try it yourself in Incognito mode:
  - Login into eClass: [eclass.srv.ualberta.ca/course](https://eclass.srv.ualberta.ca/course)
  - Open [eclass.srv.ualberta.ca/course/view.php?id=2187](https://eclass.srv.ualberta.ca/course/view.php?id=2187), you should be logged in



UNIVERSITY OF ALBERTA  
IST eCLASS SUPPORT

**Welcome to eClass for  
Students!**

If the path attribute is set too **loosely**, it could leave the application vulnerable to **attacks by other applications** on the same server.

# Basic facts

Desired properties for sessions:

- Browser remembers user
- User cannot modify session cookie to login as another user
- Session cookies only last as long as the browser is open
- Sessions can be managed (e.g., session ID can be deleted) by the server
- Sessions expire after some time