

# Anon-Pass:

## Practical Anonymous Subscriptions

Michael Z. Lee<sup>†</sup>, Alan M. Dunn<sup>†</sup>,  
Jonathan Katz<sup>\*</sup>, Brent Waters<sup>†</sup>, Emmett Witchel<sup>†</sup>

<sup>†</sup> University of Texas at Austin

<sup>\*</sup> University of Maryland

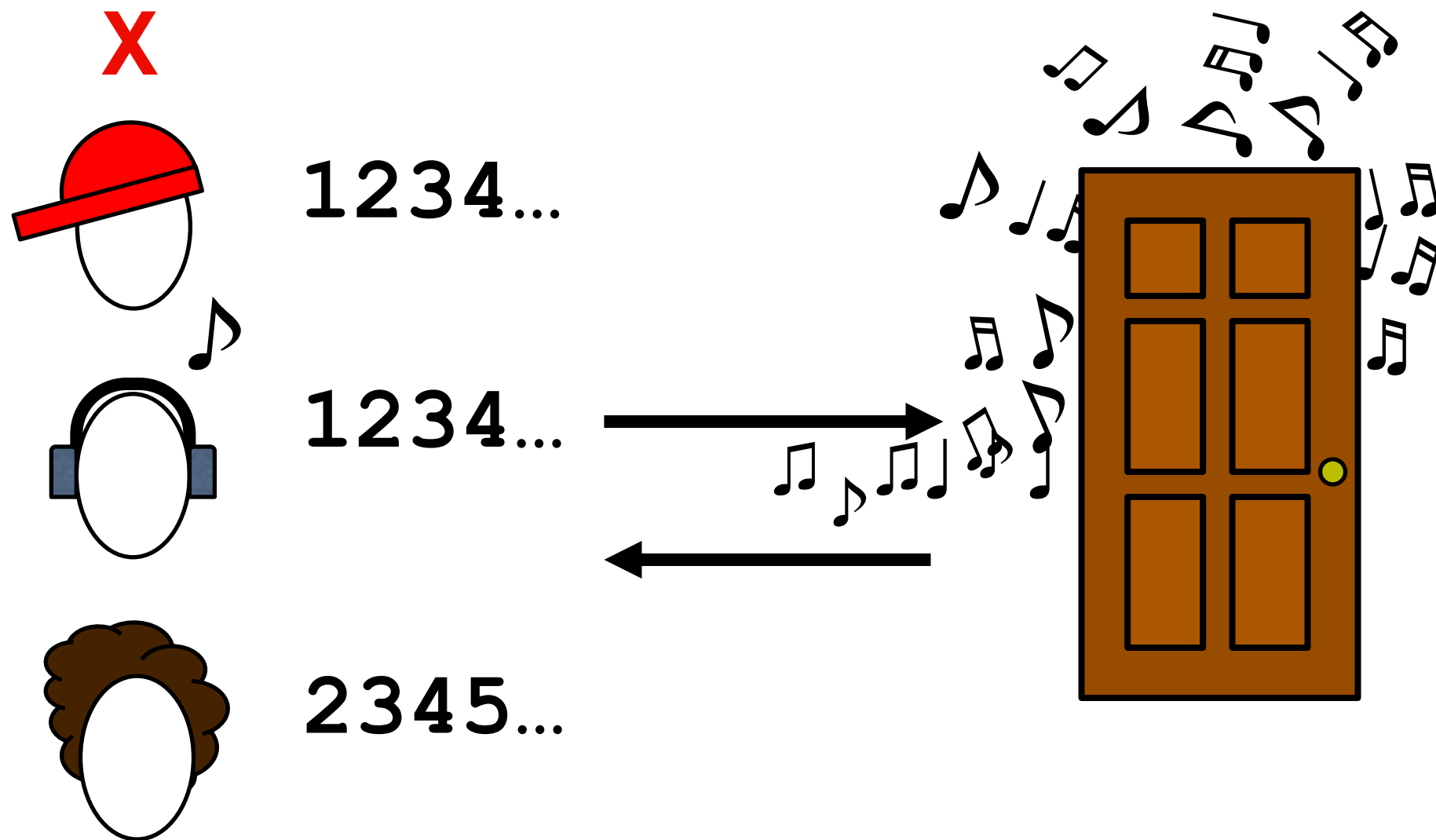
# Media Subscriptions

The Netflix logo, featuring the word "NETFLIX" in white, bold, sans-serif capital letters with a slight 3D effect, set against a solid red rectangular background.

Unlimited access subscriptions

The New York Times

# Let's build a service



**Sharing Resistance**  
(admission control)



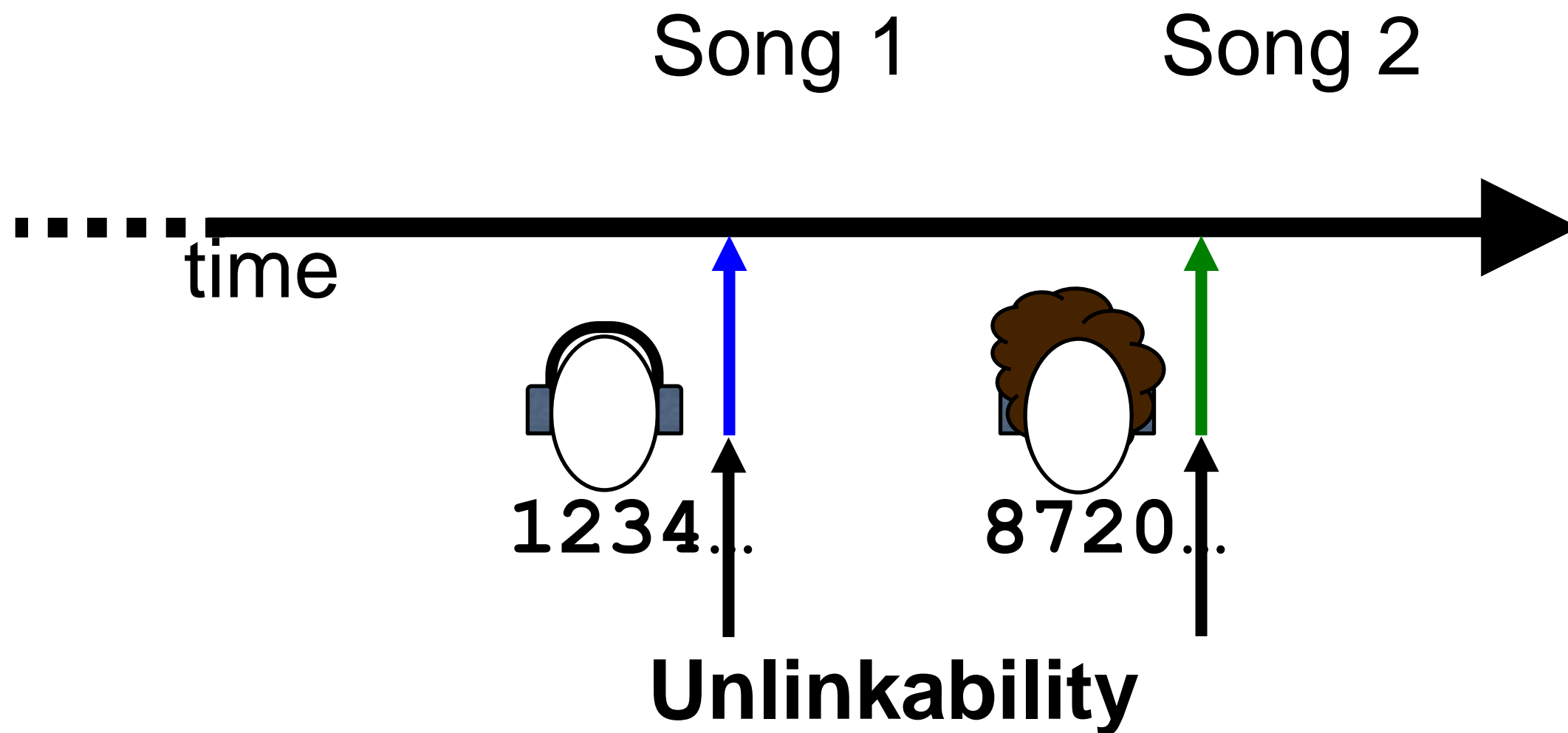
1. [Introduction](#)
2. [The information we collect](#)
3. [How we use the information we collect](#)
4. [How we share the information we collect](#)

They are collecting information about you.



# Anonymous Media

Accesses can't be correlated



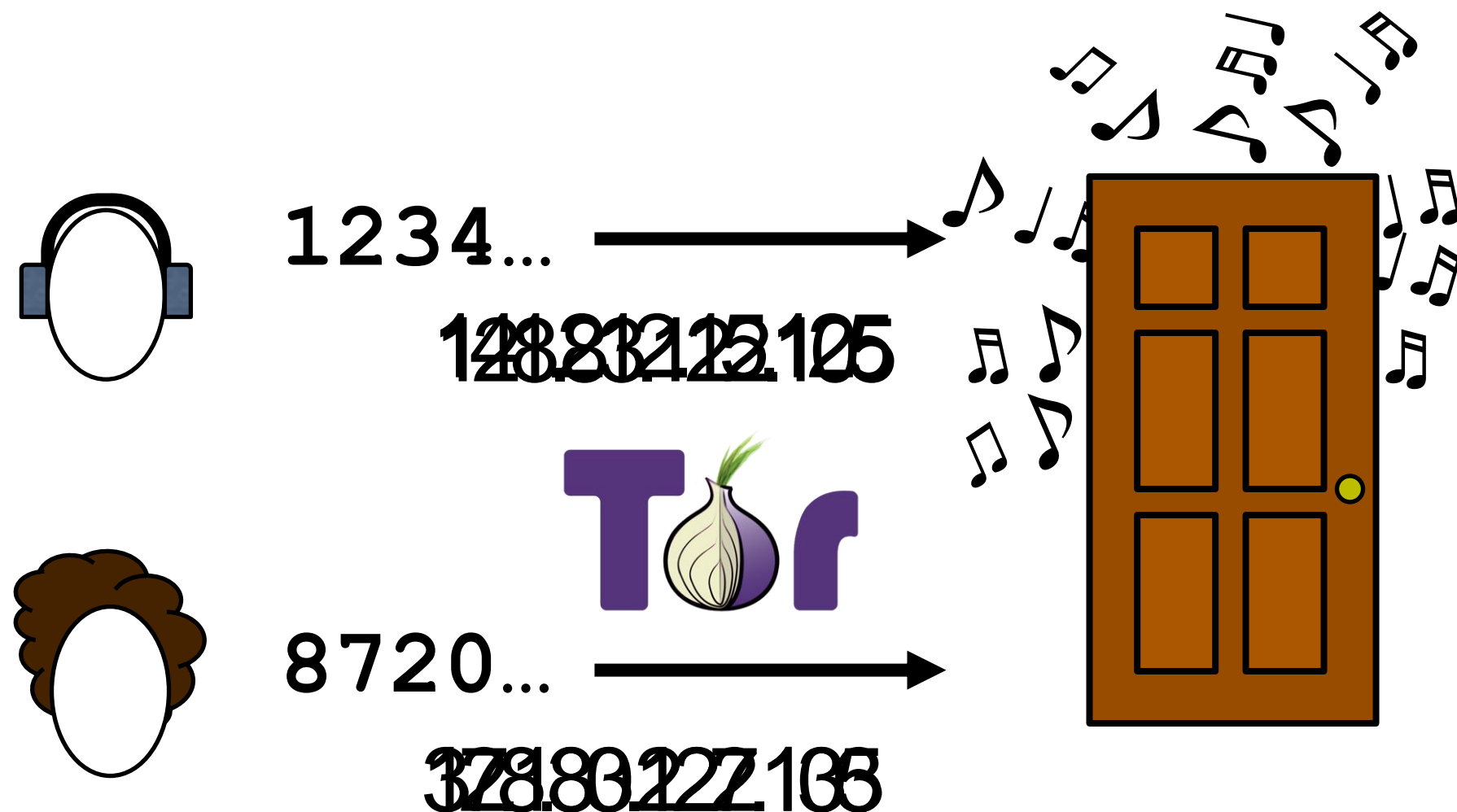
# Linked accesses could deanonymize users

Access patterns for enough time  
could help deanonymize clients

The Netflix Prize dataset  
[Narayanan, Shmatikov 2008]

Social networks  
[Narayanan, Shmatikov 2009]

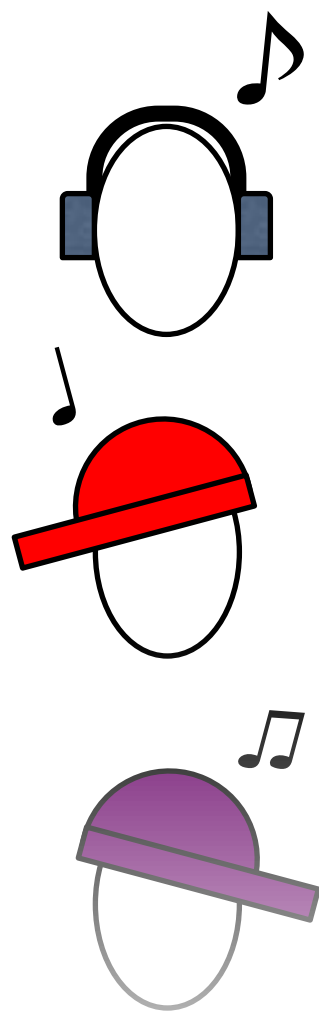
# But even if tokens are unlinkable...



We assume clients are using a network anonymity service

# Anonymous Music Service

Straw Man



1234...

8720...

7964...

8739...

1910...

2372...

3141...



**Unlinkability**

but not sharing resistance



# How do we get both?

Unlinkable Serial Transactions [Syverson et al. 1992]

Sharing resistance, unlinkability  
**but needs unbounded storage**

Anonymous Blacklisting Systems [Tsang et al. 2000]

Sharing resistance, unlinkability  
**but computationally expensive**

# And also be practical?

Unlinkable Serial Transactions [Syverson et al. 1998]

Sharing resistance, unlinkability  
**but needs unbounded storage**

Anonymous Blacklisting Systems [Tsang et al. 2000]

Sharing resistance, unlinkability  
**but computationally expensive**

## **Anon-Pass**

Sharing resistance, unlinkability, and efficiency  
Example: over 12,000 concurrent clients

# How?

How is Anon-Pass built?

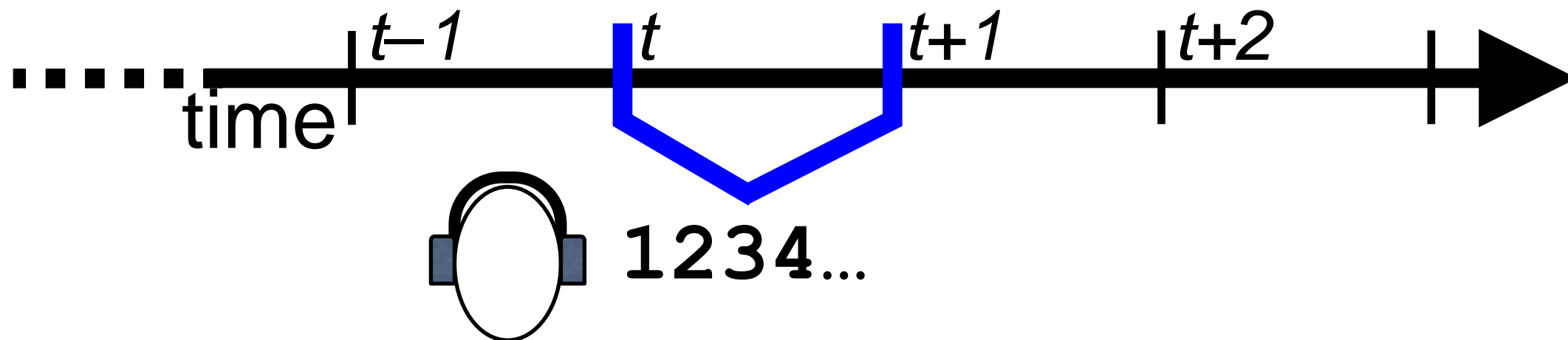
How is Anon-Pass used?

How does Anon-Pass perform?

# How is it built?

Split up time into **epochs**

Each user has a **unique** token for an epoch

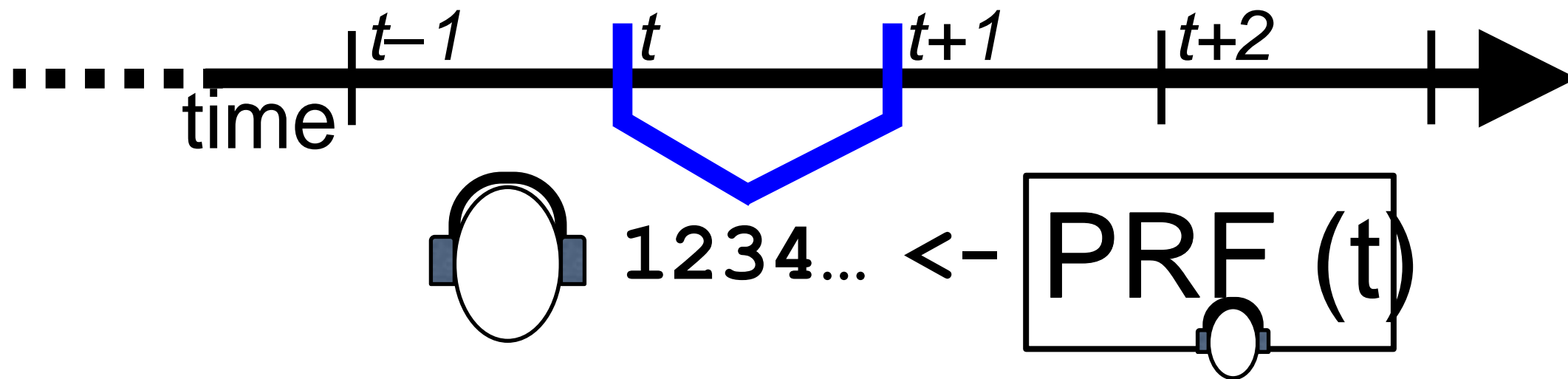


Each epoch allows a new, **unpredictable** token

# How is it built?

Split up time into **epochs**

Each user has a **unique** token for an epoch



Each epoch allows a new, **unpredictable** token

Use a **pseudorandom function** (PRF)

# High Level Protocols

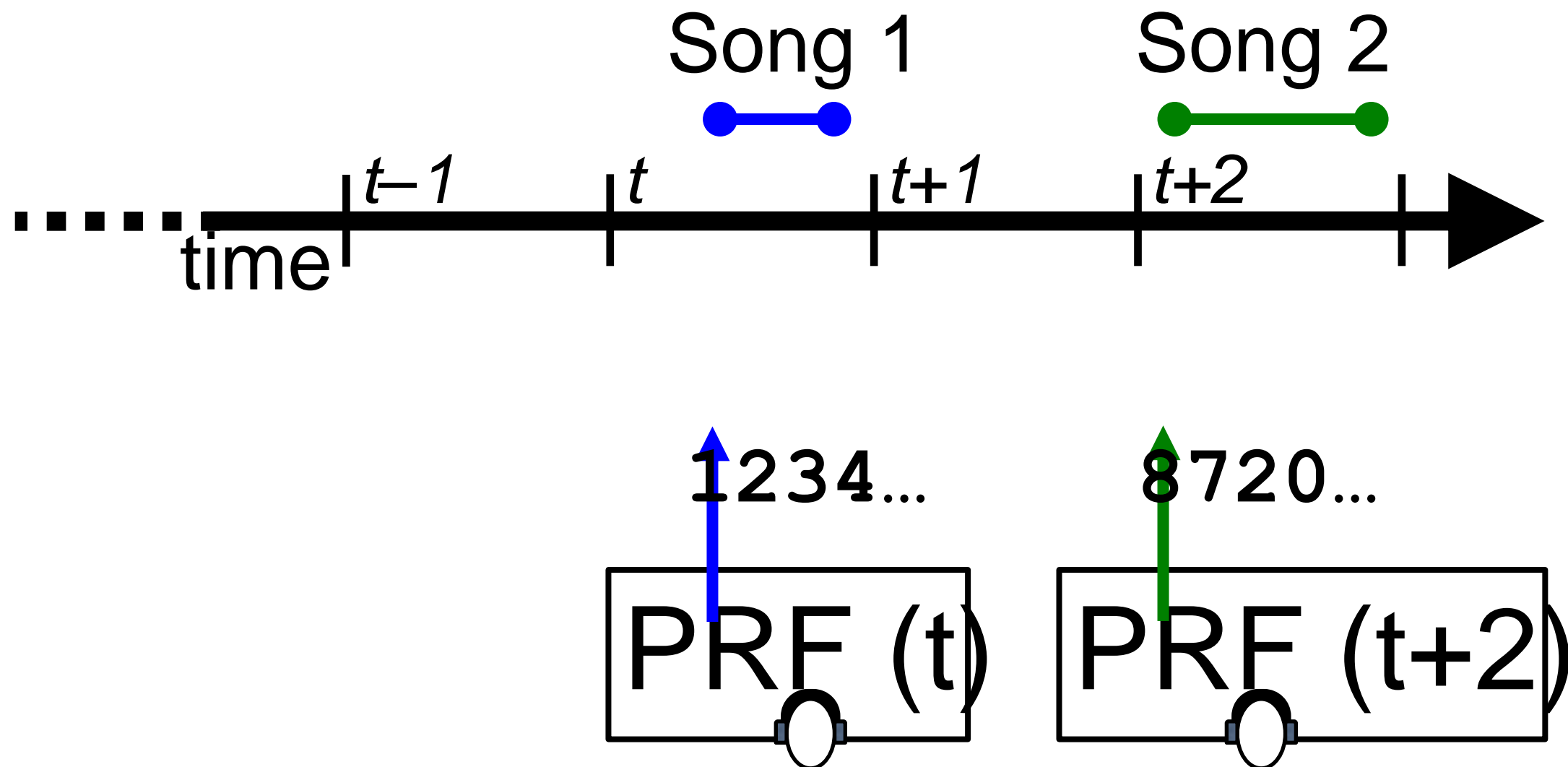
## **Register**

Get a blinded signature on a secret

## **Login**

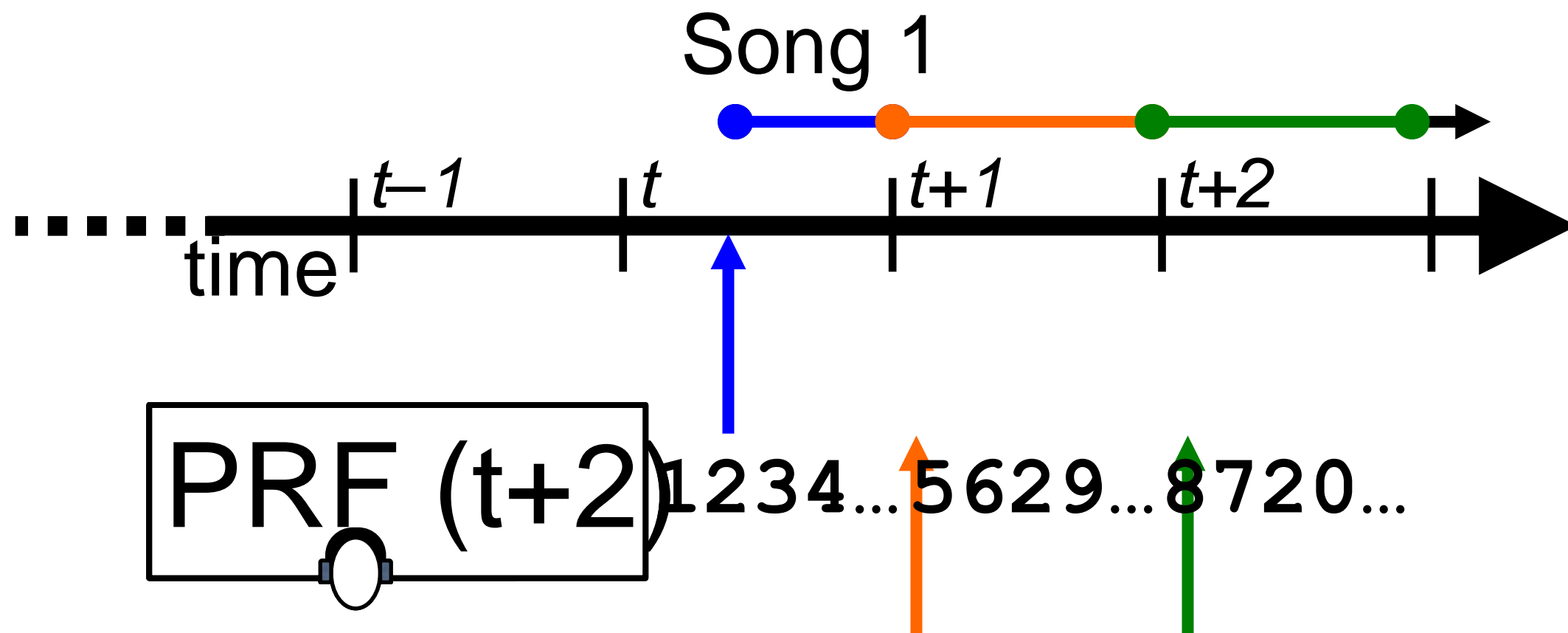
Prove the token used the signed secret  
(in zero knowledge)

# Anonymous Music Service



# Anonymous Music Service

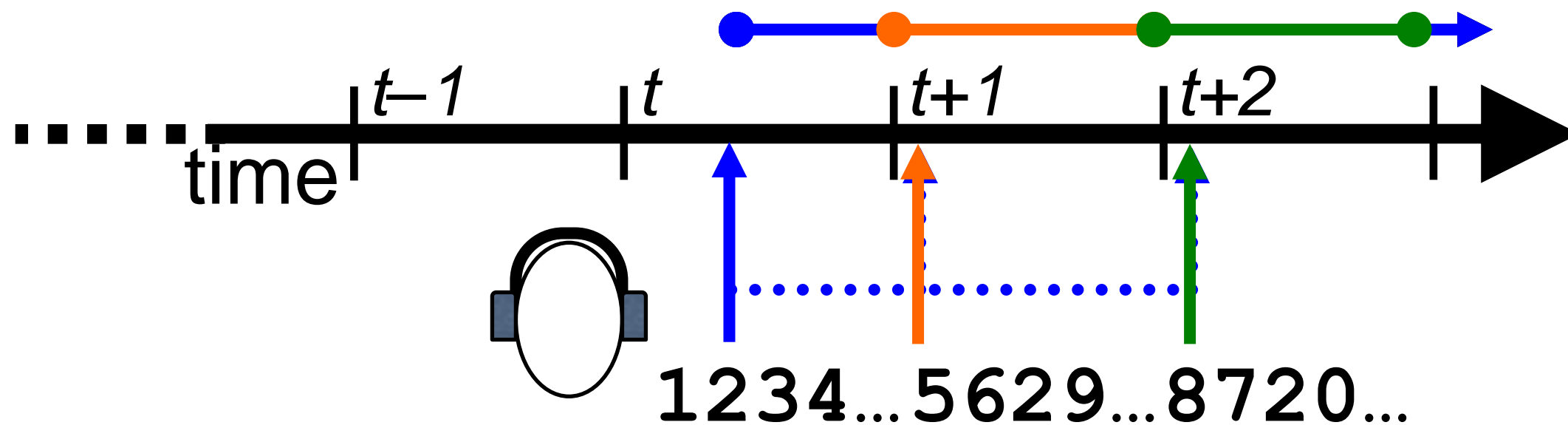
But songs don't always fit in one epoch





# Anonymous Music Service

But songs don't always fit in one epoch  
And these accesses are implicitly linked



**Conditional Linkability**

# Accesses can be implicitly linked

Baby+ 0s     The service knows when the  
Baby+15s same song is repeatedly accessed  
Baby+30s  
Baby+45s     Client is implicitly linked  
Baby+60s while accessing the same media  
Baby+75s  
Baby+90s     And unlinkability costs  
.....     the service provider  
                 (and therefore harms the system)

# Re-Up

Our way of getting  
**conditional linkability**

Prove the current token and  
the next token are linked

**Trades** unlinkability for efficiency

But the client already lost unlinkability  
while accessing the same media

# Re-Up is more efficient

Login proves you should be allowed access

Re-Up proves you logged in before

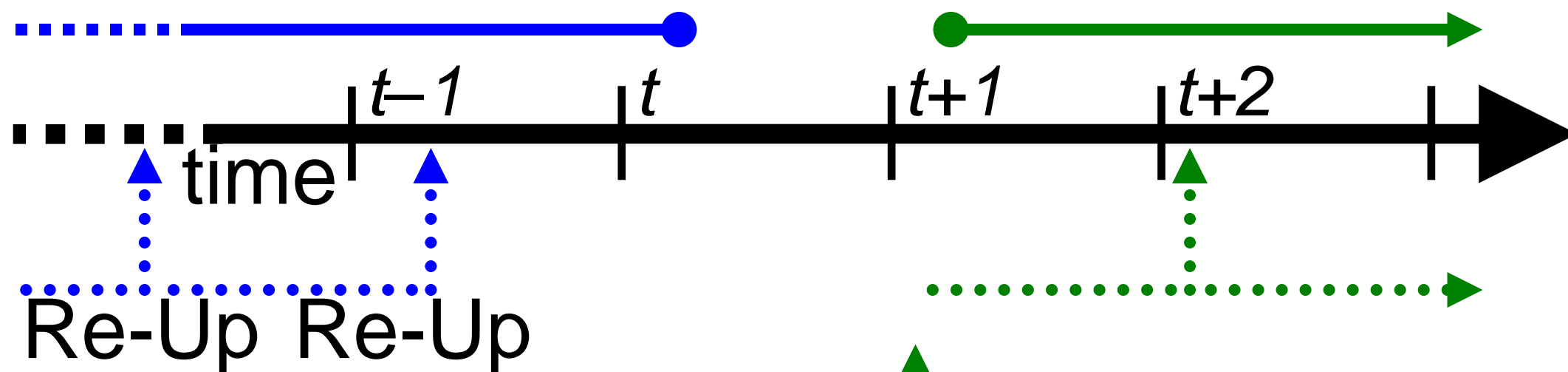
Login takes 10 expensive operations

Re-Up takes only 2

# Using Login and Re-Up

A client must Login to start a new song

And Re-Up to continue playing the same song



To be unlinkable again, the client must wait until the next epoch

# Epoch Lengths: Long vs. Short

A **short** epoch means less time to be unlinkable  
And less delay between client actions

Happy Clients

A **long** epoch means fewer client requests  
And lower server load

Happy Server

Choosing an epoch length depends on the service  
(e.g., 15 seconds for music, 5 minutes for movies)

# Re-Up helps balance this tension

Short epochs means less wait  
between unlinkable actions

Re-Up instead of Login  
reduces server load

# And Anon-Pass is formally proven

Formal proof of security holds under  
the DDHI assumption

Formal proof of soundness holds under  
the LRSW assumption

Stated and proved in the paper



# How?

How is Anon-Pass built?

How is Anon-Pass used?

How does Anon-Pass perform?

# How could it be used?

Anonymous Music Streaming

- Music download over normal HTTP

- 15 second epoch

Unlimited-use Subway Pass

- NYC's "unlimited" pass

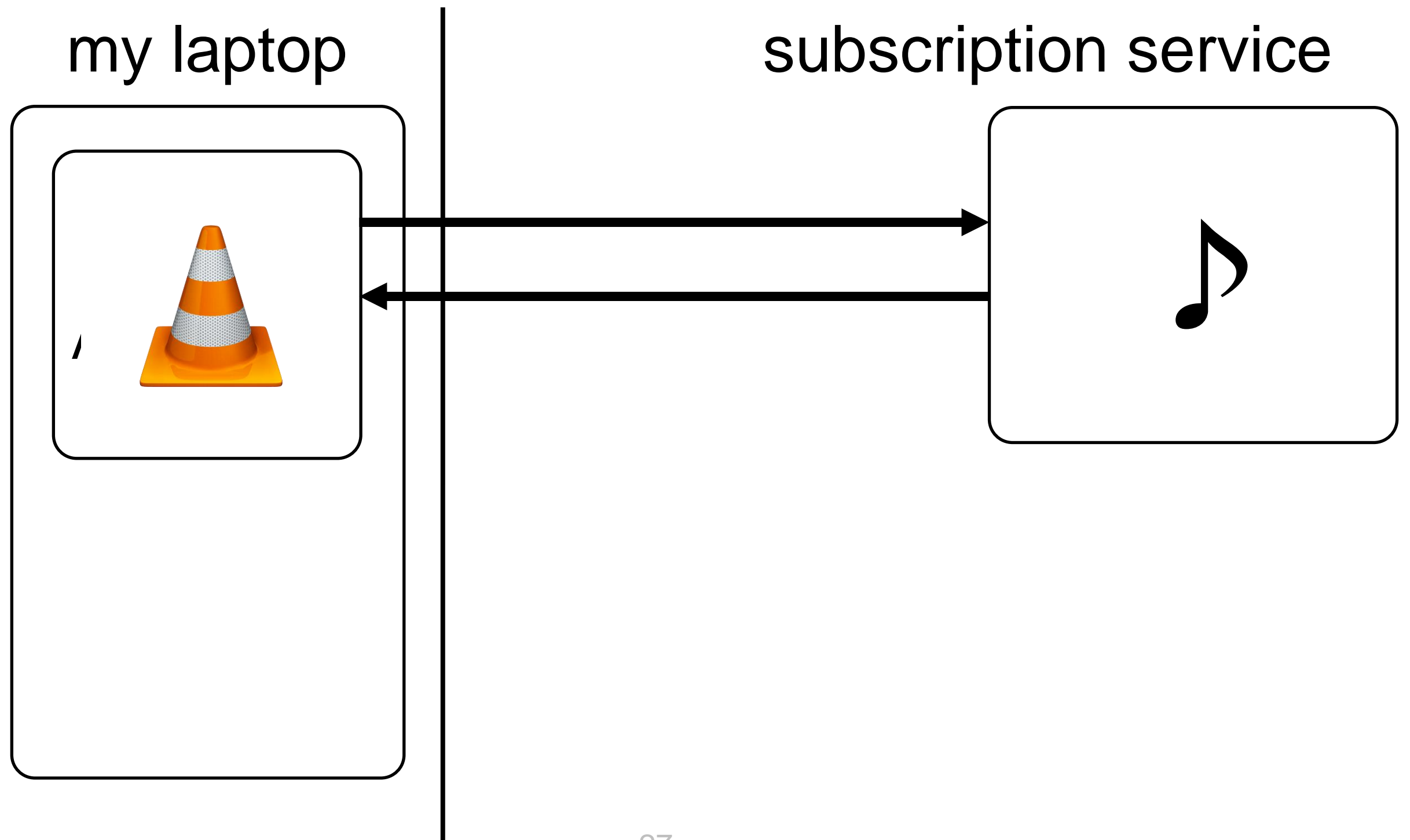
- 6 minute epoch

Account Proxy

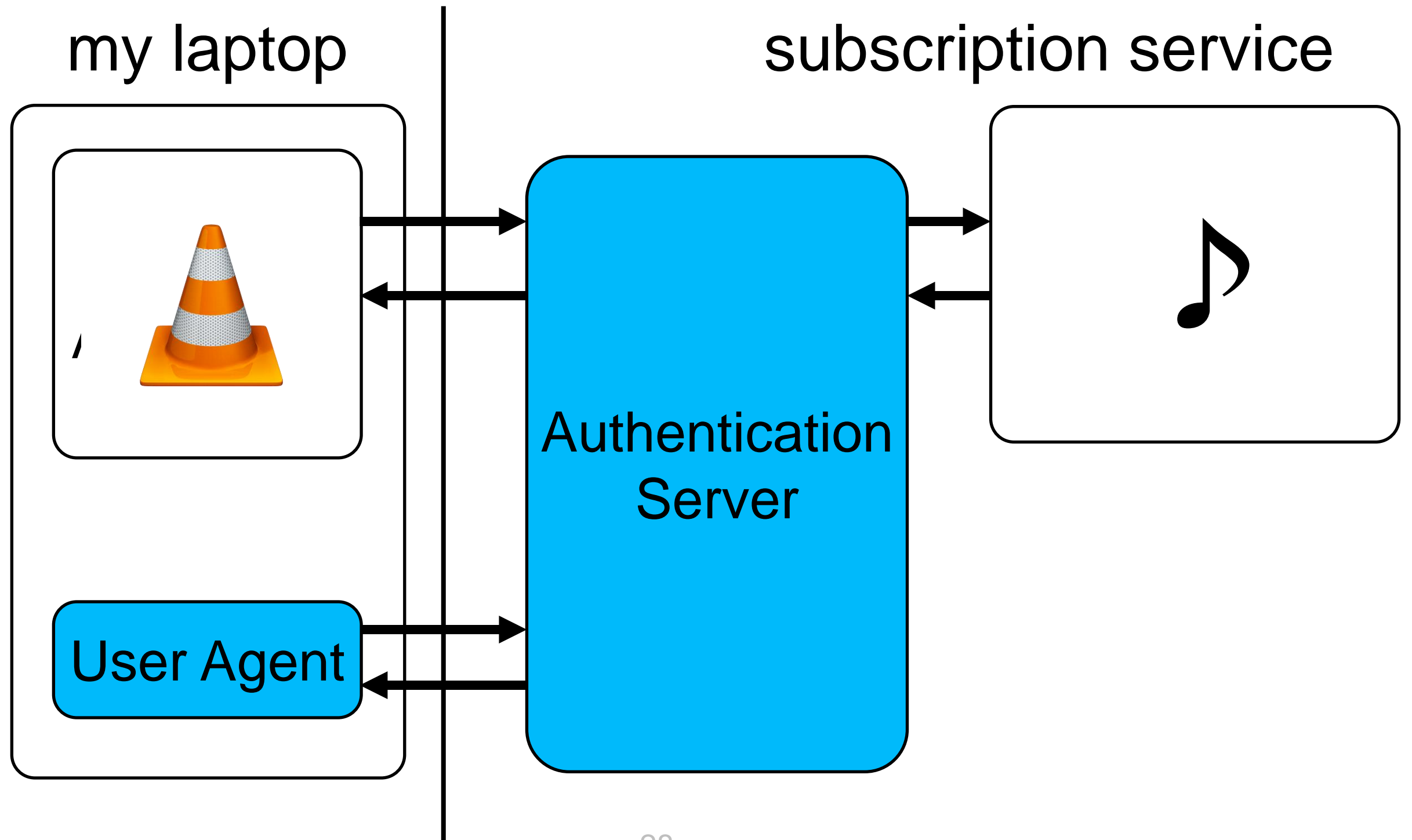
- Multiplex accounts to news sites

- 1 minute epoch

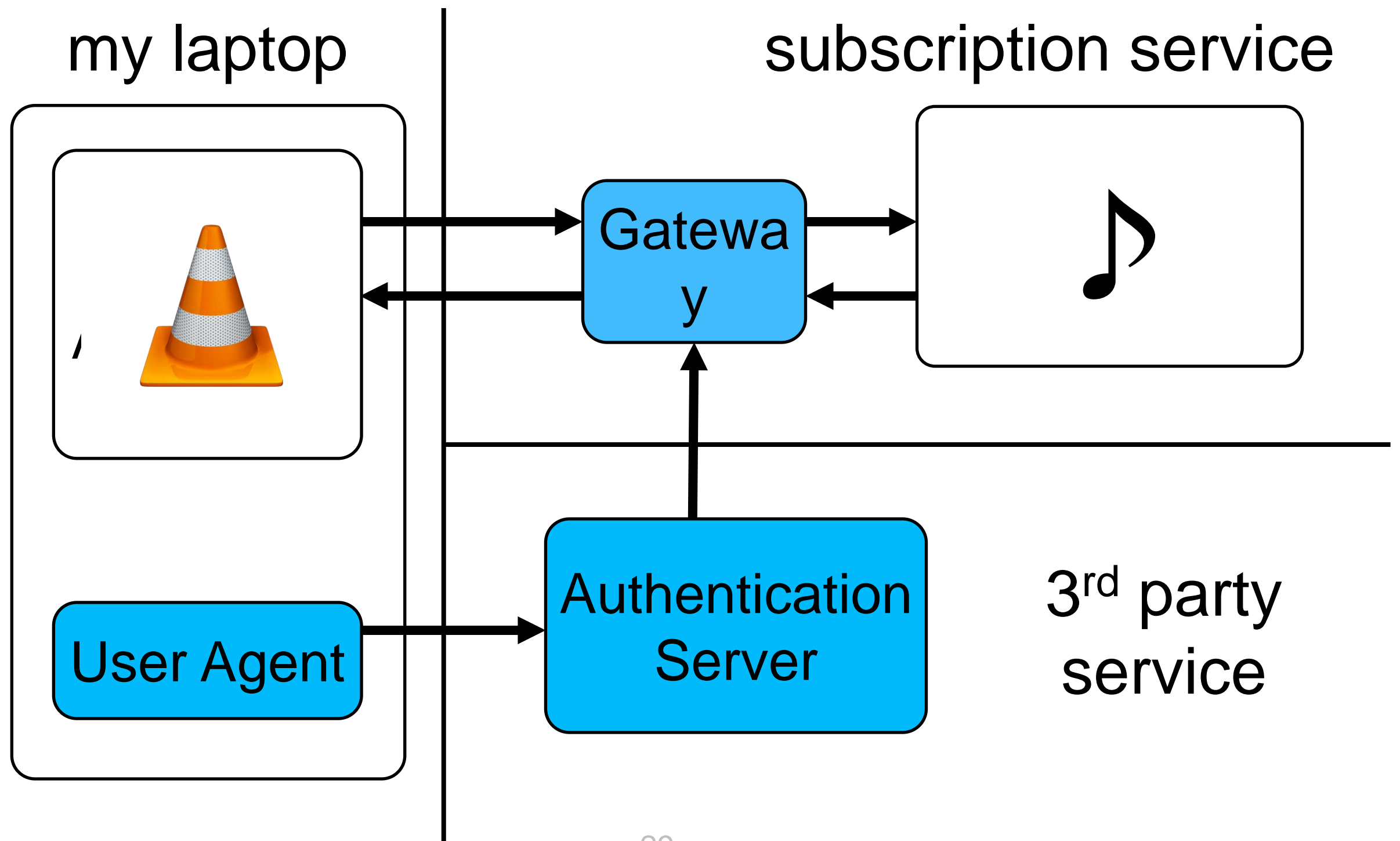
# System Architecture

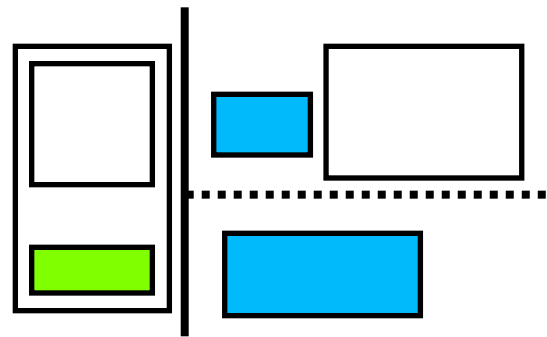


# System Architecture



# System Architecture





# User Agent

Purpose: minimize changes to client applications

Job: Create Login and Re-Up requests  
Keep the user secret secure

Modified VLC to anonymously stream (54 LoC)  
No modifications to support browsers

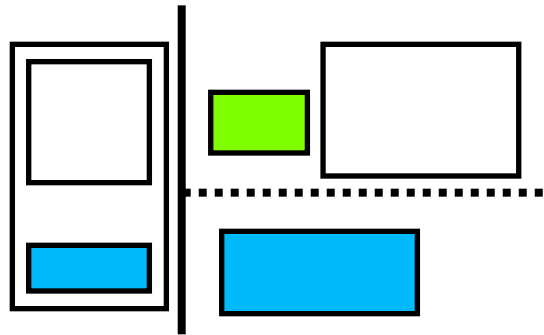
A diagram of an authentication server. It consists of a vertical rectangle on the left, a horizontal rectangle in the middle, and a larger rectangle on the right. The vertical rectangle contains a blue rectangle at the top and a green rectangle at the bottom. The horizontal rectangle is blue. The larger rectangle is white. A dotted line connects the bottom of the vertical rectangle to the bottom of the horizontal rectangle.

# Authentication Server

Purpose: enforce sharing resistance

Job: Verify tokens and token uniqueness  
Record active tokens

Runs on the service or as a 3<sup>rd</sup> party



# Gateway

Purpose: enforce access control with minimal change to existing services

Job: Prevent unauthorized access and responses  
Remove verification from the critical path

Runs on the service as a front end server



# How?

How is Anon-Pass built?

How is Anon-Pass used?

How does Anon-Pass perform?

# Evaluation Environment

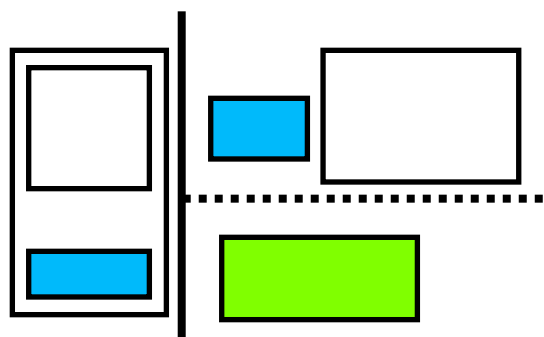
quad-core 2.66 GHz Intel Core 2 CPU

8GB RAM

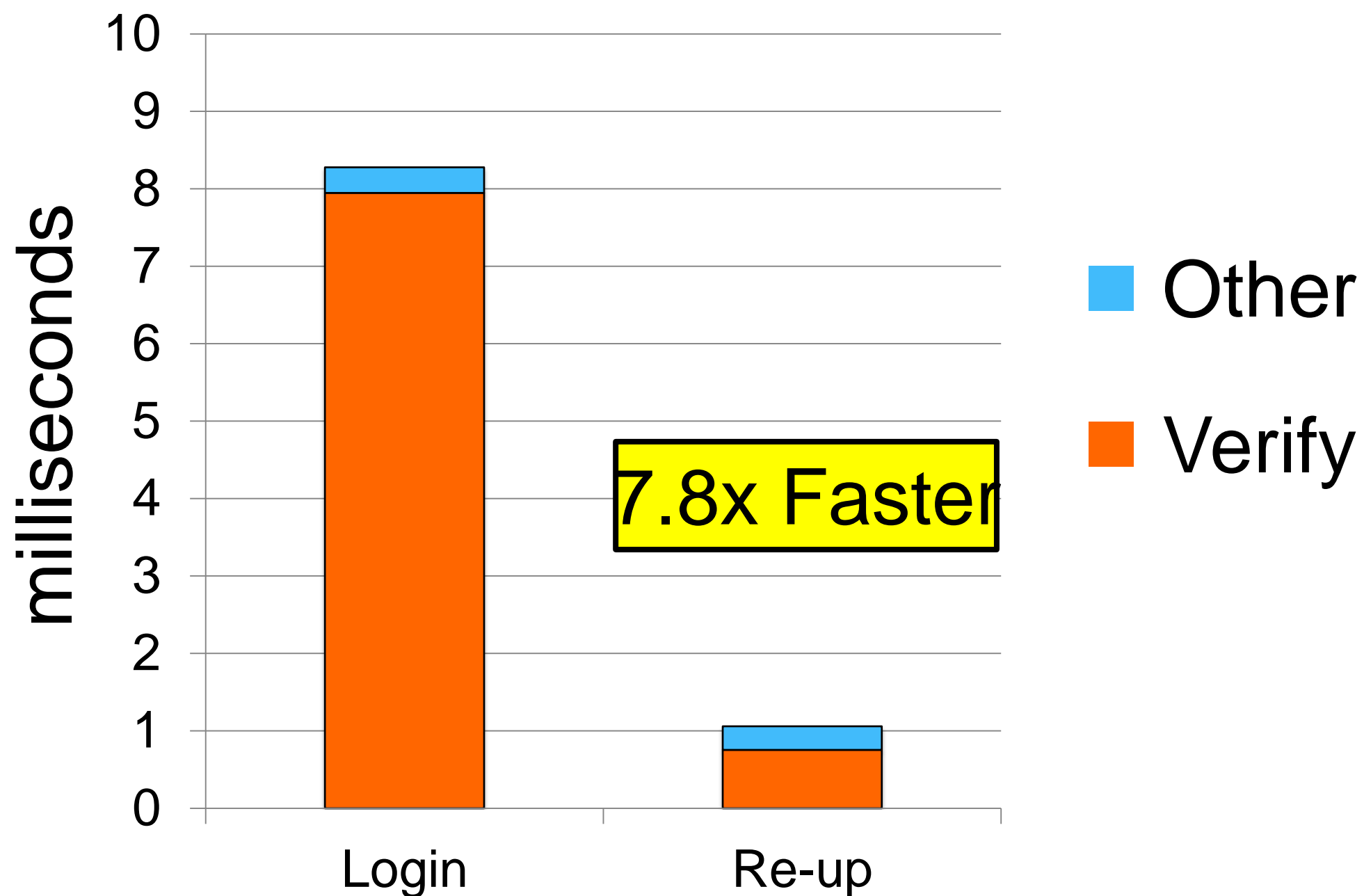
1 Gbps network

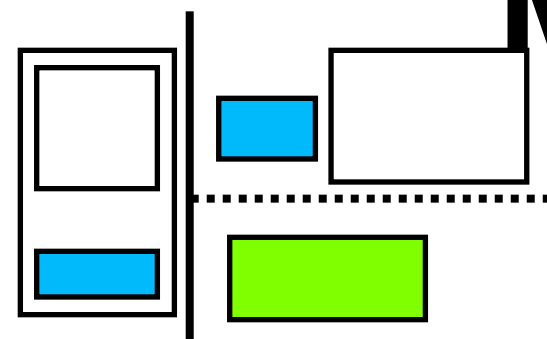
10 client machine to evaluate  
the streaming music service

An HTC Evo 3D to evaluate  
the anonymous subway pass



# Crypto Cost



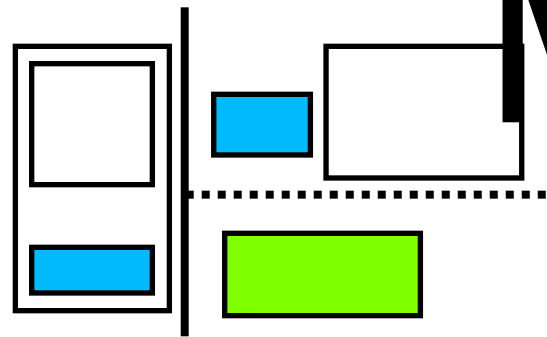


# Music Service Scaling

HTTP server to stream music  
15 second epoch

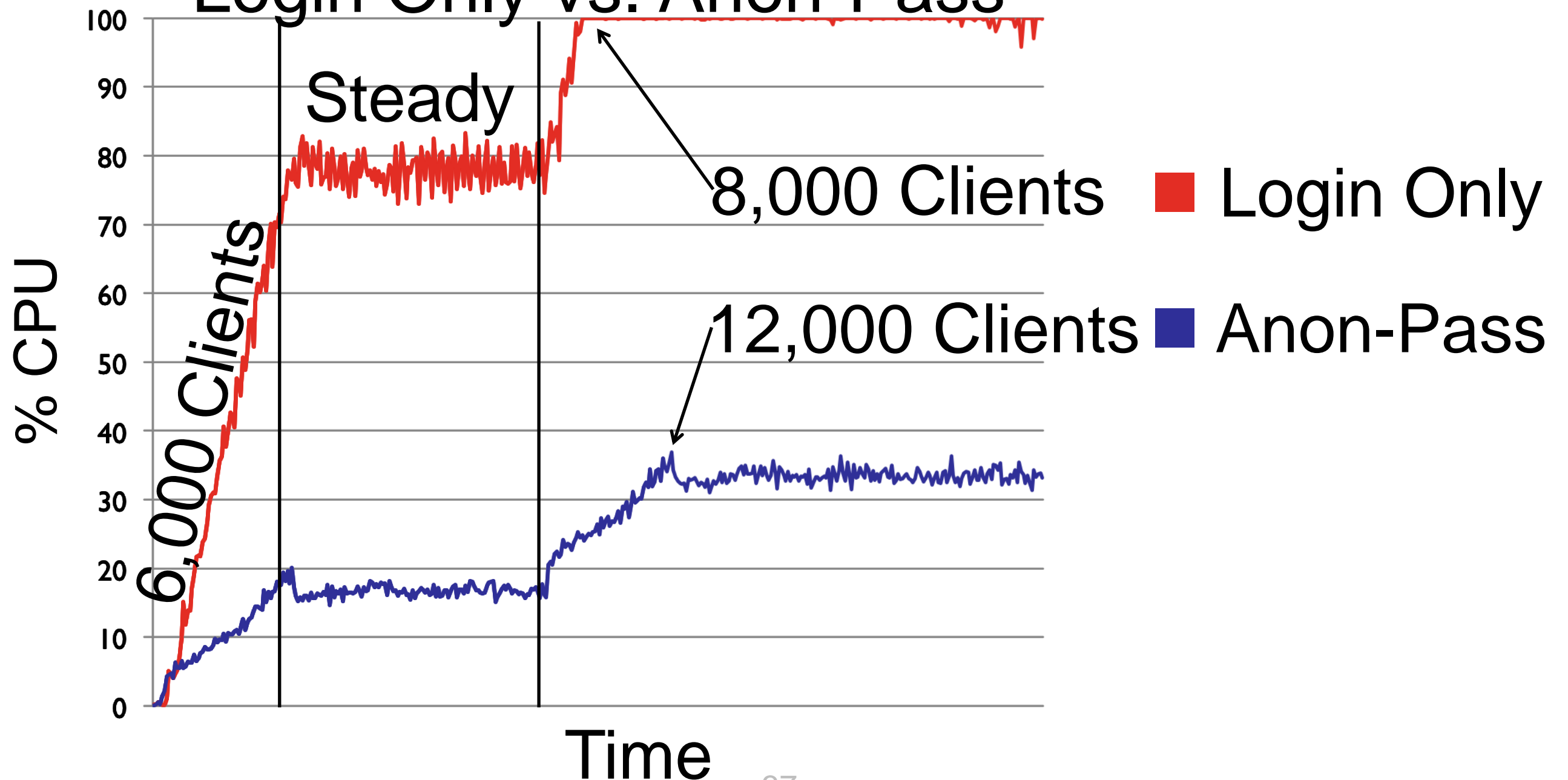
Add clients until we run out of resources

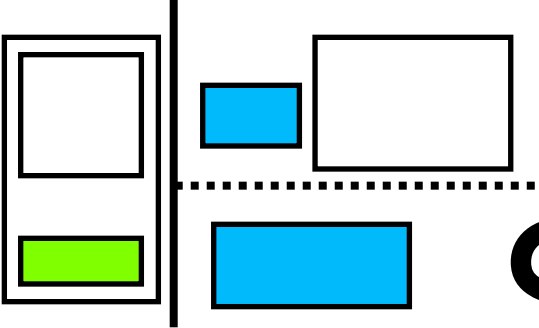
Used 10 client machines



# Music Service Scaling

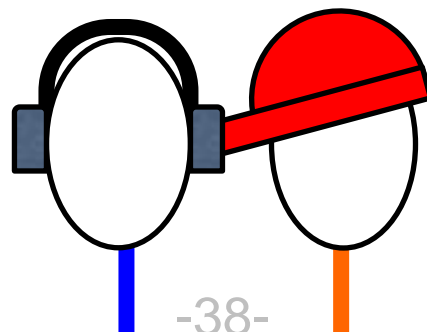
Login Only vs. Anon-Pass





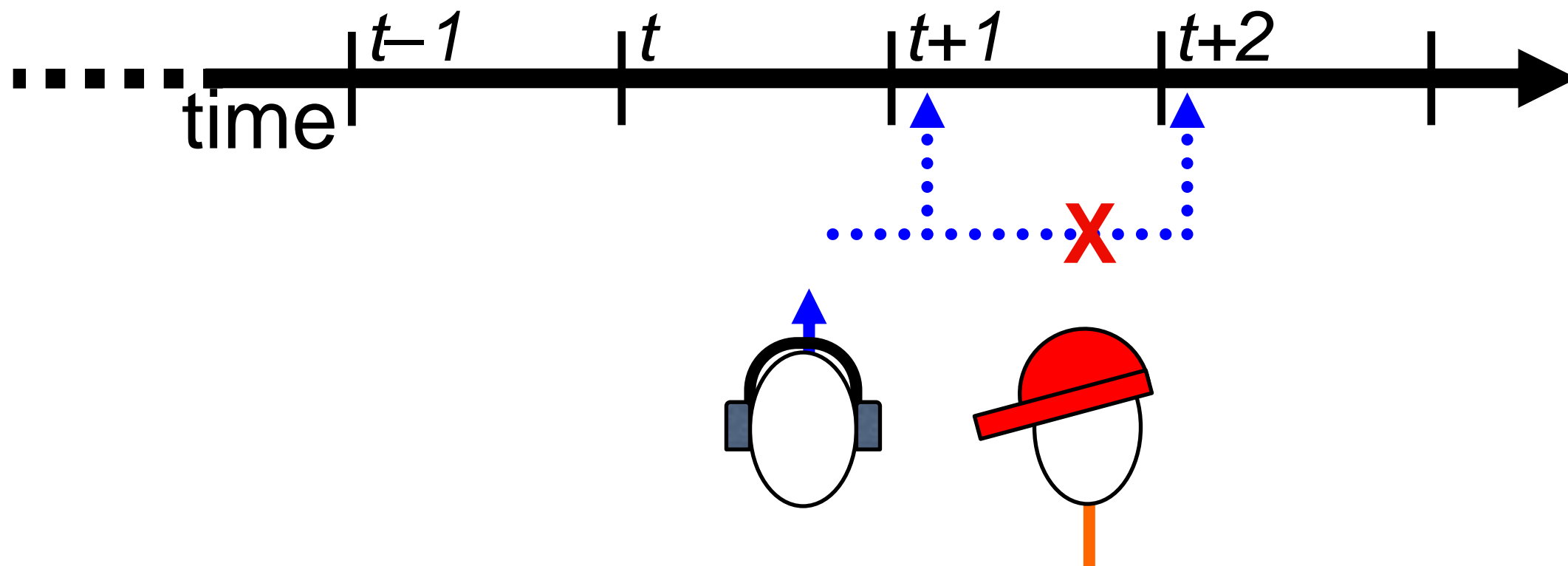
# Anonymous Subway Pass

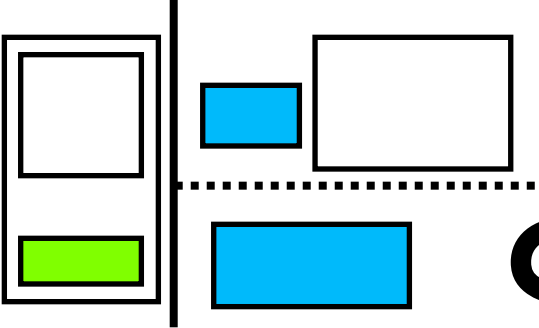
Problem: Need to rate limit between swipes  
A long epoch can simulate that timeout  
But sharing is still possible...



# Anonymous Subway Pass

Solution: Login and Re-Up at the same time  
Accesses during later epochs are linkable



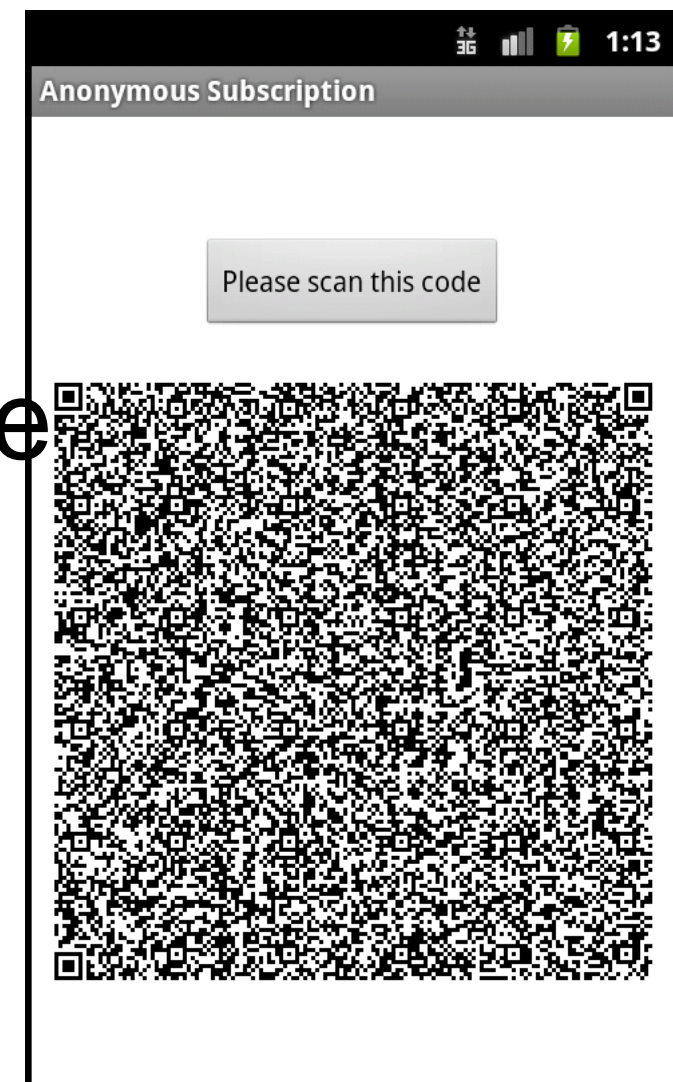


# Anonymous Subway Pass

Implemented as an  
Android application

Clients Login and Re-Up twice  
(18 minute NYC policy)

Takes only 0.2 seconds  
(on an HTC Evo 3D)





# Anon-Pass

Practical – efficient enough to scale

Flexible – works with different services

Deployable – minimizes service changes

