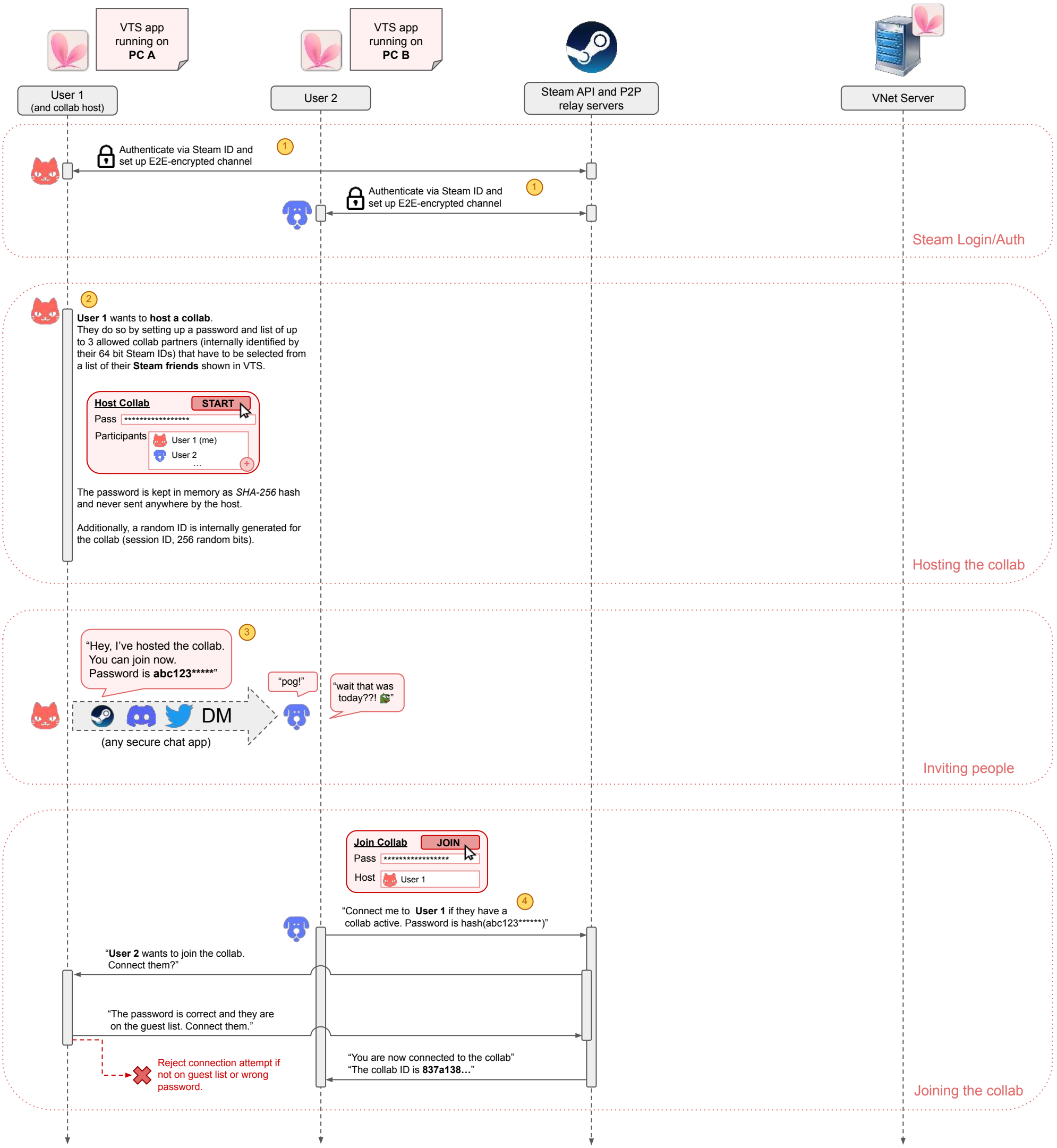

VTS app running on PC A
VTS app running on PC B
User 1 (and collab host)
User 2
Steam API and P2P relay servers
VNet Server
Authenticate via Steam ID and set up E2E-encrypted channel
1
Authenticate via Steam ID and set up E2E-encrypted channel
1
Steam Login/Auth
2
User 1 wants to host a collab.
They do so by setting up a password and list of up to 3 allowed collab partners (internally identified by their 64 bit Steam IDs) that have to be selected from a list of their Steam friends shown in VTS.
Host Collab
START
Pass **********
Participants
User 1 (me)
User 2
...
The password is kept in memory as SHA-256 hash and never sent anywhere by the host.
Additionally, a random ID is internally generated for the collab (session ID, 256 random bits).
Hosting the collab
3
"Hey, I've hosted the collab. You can join now. Password is abc123*****"
"pog!"
"wait that was today??! "
DM
(any secure chat app)
Inviting people
Join Collab
JOIN
Pass **********
Host
User 1
4
"Connect me to User 1 if they have a collab active. Password is hash(abc123******)"
"User 2 wants to join the collab. Connect them?"
"The password is correct and they are on the guest list. Connect them."
Reject connection attempt if not on guest list or wrong password.
"You are now connected to the collab"
"The collab ID is 837a138..."
Joining the collab

VTS app running on **PC A**

VTS app running on **PC B**

User 1 (and collab host)

User 2

Steam API and P2P relay servers

VNet Server

Sending Models

Load model **"Akari"**

All communication between collab partners is actually encrypted and routed over **Steam relay servers**, so clients **never directly communicate** or even know each other's IPs (see diagram above).

This diagram omits this to make the communication structure more simple.

You just loaded a new model. Do you want to share it with collab participants? 5

YES **NO**

Create random unique encryption keys for every single file in the model.

FILE_GUID:
4c904812a37ff20e077c9408bed83645

Encrypt every required file in model folder with respective random SHA key, using **256 bit AES**.

File: texture_00.png
Key: 0x0b38ae734…
AES IV: 0x83bc7195c…

Upload every file to server:
4c904812a37ff20e077c9408bed83645 **.bin**
8d22854e16c93e2ba3717e2425886618 **.bin**
d2a55998c84d9ac4e16ab865194234ef **.bin**
…

6

Server config:
- No file listing allowed.
- Rate-limited public access.
- Max. 24 hours retention policy (auto-delete after 24 hours)

(FILE_GUID)

Private file URLs:
https://the-vnet-server-url.com/<random-file-id>.bin

7 Send file info through encrypted Steam SDR channel.

File: texture_00.png
Key: 0x0b38ae734…
AES IV: 0x83bc7195c…
File ID: 4c904812a37…

"File info received"

Distribute file info to any other peers through encrypted Steam SDR channel.

Access model files for local display: https://the-vnet-server-url.com/4c904812a…

Download into memory (RAM, **not HDD**)

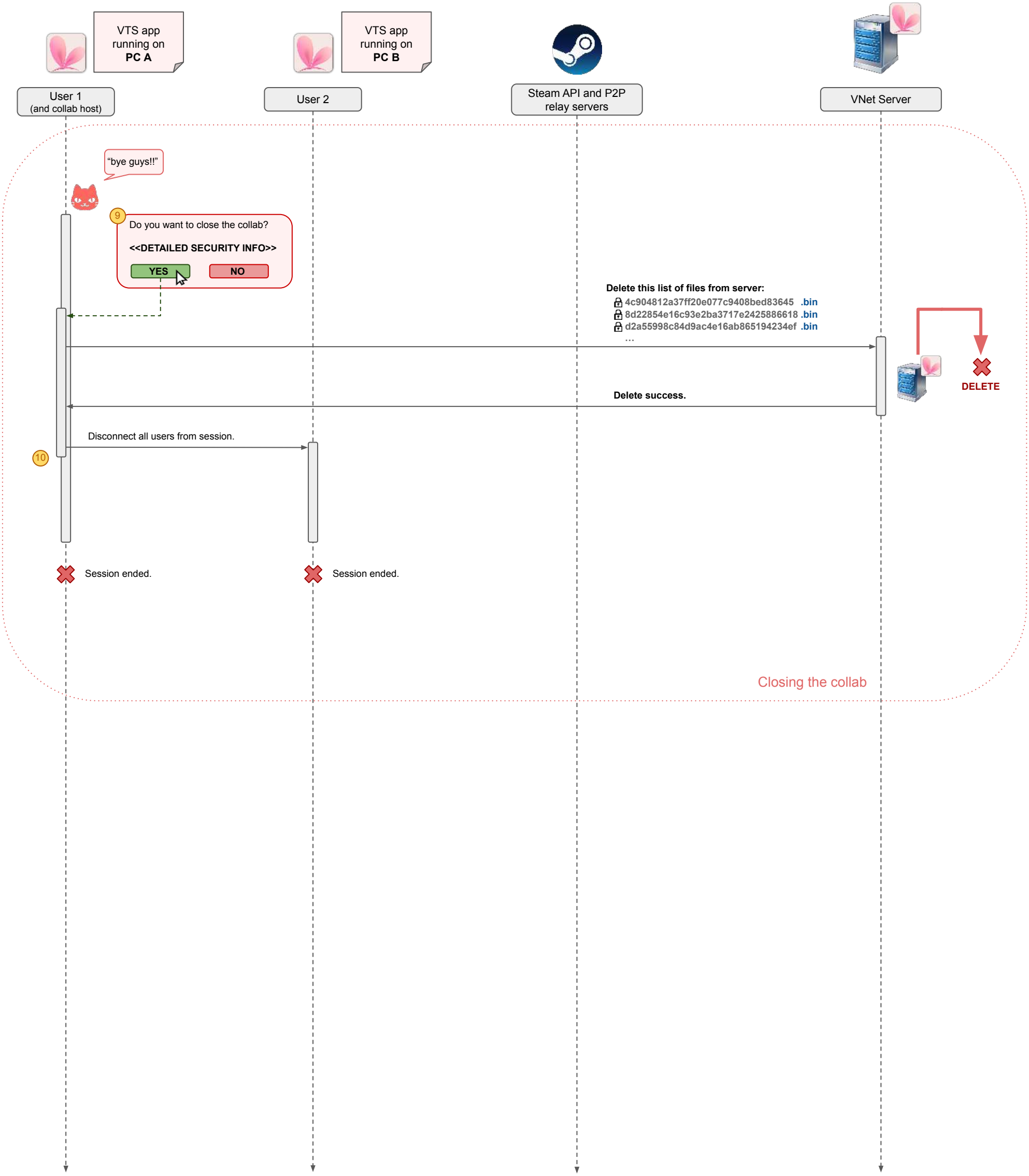
8

Model: Akari

Decrypt all model data files in memory and load model from memory.

Model files are **never saved** on collab partner PCs.

Sending Models


VTS app running on PC A
VTS app running on PC B
User 1 (and collab host)
User 2
Steam API and P2P relay servers
VNet Server
"bye guys!!"
9
Do you want to close the collab?
<<DETAILED SECURITY INFO>>
YES
NO
Delete this list of files from server:
4c904812a37ff20e077c9408bed83645 .bin
8d22854e16c93e2ba3717e2425886618 .bin
d2a55998c84d9ac4e16ab865194234ef .bin
...
DELETE
Delete success.
Disconnect all users from session.
10
Session ended.
Session ended.
Closing the collab

IDs, hashes and encryption

Unique IDs in VTube Studio typically use GUIDs. Specifically, the GUIDs are Version 4 Universally Unique Identifiers as described in RFC 4122, Sec. 4.4.

For random encryption keys, the *.NET RandomNumberGenerator* is used to create cryptographically strong random numbers. For example, this is used to create the random 256 bit keys to to encrypt files using *AES-256* (first encryption step) before uploading them to the VNet server.

These keys and their initialization vectors are then shared with fully authenticated collab participants.

For hashing, VTube Studio generally uses salted *SHA-256* hashes wherever applicable.

Where are model files stored? Who are they shared with?

Note: For a more detailed explanation, please refer to the step-by-step guide further below.

When you join a collab or load a new model, you will be asked if you want to share your model. If you don't want to share it, nothing happens. If you do want to share it, a random 256 bit encryption key is generated for all relevant files in your model and the *gzip*-compressed files are encrypted with their respective key. The keys are never revealed anywhere on the UI, so you cannot accidentally leak them. All files will then be encrypted again with another 256 bit key that is derived from the hashed session password.

Each file will also get its own random ID, which will be used as its filename when uploading.

The files will then be individually uploaded to the VTube Studio server. The path for a file will be something like this:
https://<vnet-server-url>/vnet-file-api/**<random-file-id>**

The VNet backend is unauthenticated (technically it is authenticated, but only using a static password to prevent random scraper bots from messing with it). Per-IP rate limiting and DDOS-protection is applied using CloudFlare.

The random file IDs are never shown to you or other collab participants, so they cannot accidentally leak. To put it simply, this is a very long and random URL that cannot be guessed by outsiders. Listing all files on the VNet file server is also not possible, so the list of files is only managed by VTube Studio locally in the background.

After the upload is done, the random file IDs and their encryption keys are automatically shared by VTube Studio with the other collab participants in the background over the end-to-end encrypted peer-to-peer channel provided by Steam.

Even if someone had full access to all files (which they don't), they would not be able to decrypt them without the per-file encryption keys that are only available within the collab. The keys are only available within the memory of collab VTS clients, so once the collab ends, all copies of the keys are lost. Even with those keys, an outsider would not be able to fully decrypt the files since a second layer of encryption is applied using the session password, so an attacker would need that as well.

Collab clients download and decrypt files into memory. **At no point do clients actually save any of the model files on their hard drive**.

When are the files deleted?

When the collab ends, the collab host automatically triggers the deletion of all files shared within the session. When this happens, all files are instantly deleted from the VTube Studio server.

Finally, all files have a global expiration time of 24 hours set as file retention policy. That means that any files are guaranteed to be deleted after 24 hours should anything go wrong (for example if VTube Studio crashes for the host before the file delete request can be sent).

How secure is this? How could someone access my model files?

To access the files, someone would need to know where they are stored (file IDs), what the encryption keys are (file keys) and what the session password is.
Given that information, a hacker could create a tool to access specific shared files. This information is stored by VTube Studio in memory on the PCs of all collaboration participants (but never shown in the logs or UI).

So in simple terms: A hacker who has full access to the PC of one of the collab participants (or a malicious collab partner) could theoretically run a tool to extract the keys, file IDs and collab ID from the VTube Studio process memory. Using that, they could download, decrypt and save the models.

As unlikely as that may be, this is a risk that is unfortunately unavoidable and this is also why **you should not collaborate with people you don't trust**. When you collaborate with other people using this tool or other tools, it is your responsibility to make sure your PC is clean and protected.

Generally, using video-based collaboration solutions may be preferable for many users.

Note: I (VTube Studio developer) have admin access to the VNet server backend, so I have access to the list of encrypted model/item files while a collab is active, but none of the info required to decrypt them or who owns them, so I do not have access to your models. I also do not see any information about who has uploaded a certain file. In the current implementation, I cannot see the IP, Steam ID or anything else.

Can anyone join/hijack the collab?

No. All collab partners have to be manually allowed/added by the host from their Steam friends list. Also, they need the collab password. The host may remove people from the collab at any time.

Also, all participants are fully authenticated with their Steam ID so when someone joins the collab, you can be 100% sure that they are who they are. Unless their Steam account got hacked of course, which is why you should always use 2-factor-authentication (you can easily set that up in the Steam settings).

What other data is shared? How do items work in collab sessions?

Other than models, items are also shared the same way. This includes Live2D items. There is a per-participant Live2D parameter limit for Live2D items (currently set at 100 Live2D parameters).

Model animation data is shared directly via the encrypted P2P connection from the clients to the host. The host then distributes that data to the clients.

Could my IP get leaked using this?

VTube Studio uses the private *Steam Datagram Relay* network to route your traffic between P2P collab partners. That way, none of the collab participants will see your IP and neither will it be shown on the UI.

How performant is this? Do I need a strong PC and good internet?

Rendering and animating multiple complex Live2D models may require a good CPU. But essentially it's the same as loading multiple models in one instance of VTube Studio via *VTube Studio Live2D Items*.

Since all models are stored in memory and never saved on the hard drive of collab partners, loading multiple big models may also require a significant amount of memory: if the downloaded encrypted model files are 50 MB in total, it will increase the VTube Studio memory usage by at least 50 MB when the model is downloaded. Same goes for items.

As for the internet connection, it depends how fast you want to send model animation data to the other collab participants. At 30-60 FPS, you can expect VTube Studio to require about 30-60 kb/s of bandwidth up/down. For the collab host, the required bandwidth might be more in the 100-300 kb/s range. Mainly this depends on the complexity of the used models (amount of Live2D parameters).

Step-by-step explanations:

1

When using the *Steam API*, every user (or “client”) will be fully authenticated. When you’re logged in with your Steam account and a game connects to the *Steam API*, all traffic will be fully end-to-end encrypted and there is no way to circumvent authentication (i.e. communicate with Steam servers without logging in or pretending you’re a different user).

The newer Steam networking system (called *SteamNetworkingSockets*) offers games/apps a way to build on this by sending peer-to-peer (P2P) messages between authenticated clients just by using their Steam ID, which means the actual IP of those clients will never be leaked to other clients because all traffic is routed through the private Steam network (*Steam Datagram Relay / SDR*).

This also means that all traffic sent P2P between clients via Steam is fully/strongly E2E-encrypted and authenticated, so if you receive a P2P-message from a client, it is guaranteed to be from the correct sender. To quote the Steamworks Documentation:

“Strong encryption and authentication. When a player connects, you can be sure that if a certain SteamID is authenticated, that someone who has access to that person's account has authorized the connection. Eavesdropping / tampering requires hacking into the VAC-secured process. Impersonation requires access to the target's computer.” - https://partner.steamgames.com/doc/api/ISteamNetworkingSockets

2

To host a collab, you have to provide two things:

1. A password
    - VTS will enforce various rules to make sure the password is reasonably safe (minimum length and complexity).
    - The password will be hashed using *salted SHA-256* when the collab is started. You cannot change it without restarting the collab.
2. A list of allowed/approved Steam IDs
    - The host must add those IDs from a list of their Steam friends. The actual IDs are never shown on the UI, so they can’t be accidentally revealed in case you don’t want your Steam ID to be public (although they are generally not really confidential).
    - The host cannot add anyone other than their Steam friends.
    - The list is stored as plaintext in memory.

When those two things are in place, the host can “start” the collab using a button on the UI. This DOES NOT publish any info about the collab/room/etc. to the Steam servers. It simply starts a local server that other people can now theoretically connect to through Steam P2P networking if they know your Steam ID.

This also creates a unique ID for the collab (“SESSION_GUID”). This key is never shown to users but sent to all successfully connected collab clients in the background.

VTube Studio does not use Steam Matchmaking or Lobbies/Rooms, so there is no way to send “invites” to other people using the normal Steam game invite mechanism. This makes it possible to run this collaboration setup alongside any other Steam multiplayer game that uses those invite/matchmaking/lobby features. To do this, make sure you start the game you want to play AFTER VTube Studio so Steam marks it as the “currently played game”.

3

There is no built-in way to invite people or pull them into the collab directly somehow.
If you want to invite someone, make sure you’ve added them to the list of allowed people (must be the Steam account they use for the collab). Then, send them the password in some safe way, for example in a private Discord chat.

Keep in mind that even if the password leaks somehow, nobody can join the collab if they aren’t on the guest list that the host set up. The password only provides an additional, redundant layer of security.

4

When **user B** wants to join the collab, they just have to select **user A** from a list of their Steam friends, put in the password and click **“Join”**.
This will trigger VTS to send a connection request to the **Steam ID of user A**.

Please note that all of this communication is routed through the Steam Datagram Relay, so the **IPs of the clients are never exposed** to each other and **all communication is fully encrypted in all directions**.

If **user B** is on the **guest list**, the host (**user A**) will check if the password is correct. If so, the host will accept the connection and **client B** will be part of the collab. Otherwise, the connection is rejected instantly.

5

The newly joined collab participant (**user A**) now loads a model. The user is asked if they want to load the model since it will be shared with all other collab participants. If the user declines, nothing happens.

If the user confirms, one random 256 bit encryption key and an unique ID will be created for each file required for the model. The files will then be compressed and encrypted using AES-256 with the generated key. Files will then be encrypted again using a key calculated from the hash of the session ID combined with the hashed password and uploaded.

The path for a file will be something like this:
https://<vnet-server-url>/vnet-file-api/**<random-file-id>**

Files are generally publicly accessible if you know their random ID, but file-listing is not allowed by the server.

Step-by-step explanations:

6. VNet server access is generally unauthenticated and only protected with a static hard-coded password. Some basic checks and rate limiting are applied server-side to prevent abuse.

   Most importantly, the server does not allow listing files. That means nobody can just list all available files. Without knowing the specific file IDs, you cannot access any files.

   Finally, the server enforces a file retention period of 24 hours. After a file is older than that, it will be automatically expired and deleted from the server.

7. After files are uploaded, all file info is sent to collab partners over the encrypted P2P channel. For a given file, that includes the filename, random file ID (to find it in on the VNet file server), AES decryption key, and AES initialization vector.

   Clients that load models send this info to the host and the host re-distributes it to all other collab clients.

8. Once a client receives info about newly uploaded files, they are downloaded into memory. They are never stored on the hard drive. When all files for a given model are downloaded, the model is loaded directly from memory into the scene.

   This is implemented in a way that does not require any files to ever be present on the hard drive of the client.

9. When the host closes the collab, their VTube Studio will request the VNet server to delete each file shared in the collab. This isn't really that important since all files are strongly encrypted and will be auto-deleted after 24 hours anyway (this is also the hard time limit for collabs).

10. When the host closes the session, all other users will be instantly disconnected. The host can now start a new session, but all models/items would have to be shared, uploaded and downloaded again.

    Non-host clients can leave and re-join the session at any point. As long as they don't close VTube Studio or join a different session in the meantime, the models/items from the old session will be retained in memory and can be instantly loaded into the scene again when you re-join the old session. This is especially important because you could get disconnected for a few seconds due to connection hiccups.

    If you get frequent random disconnects, please check the official Steam guide on how to fix this:
    https://help.steampowered.com/en/faqs/view/1F39-DCB4-FF28-5748