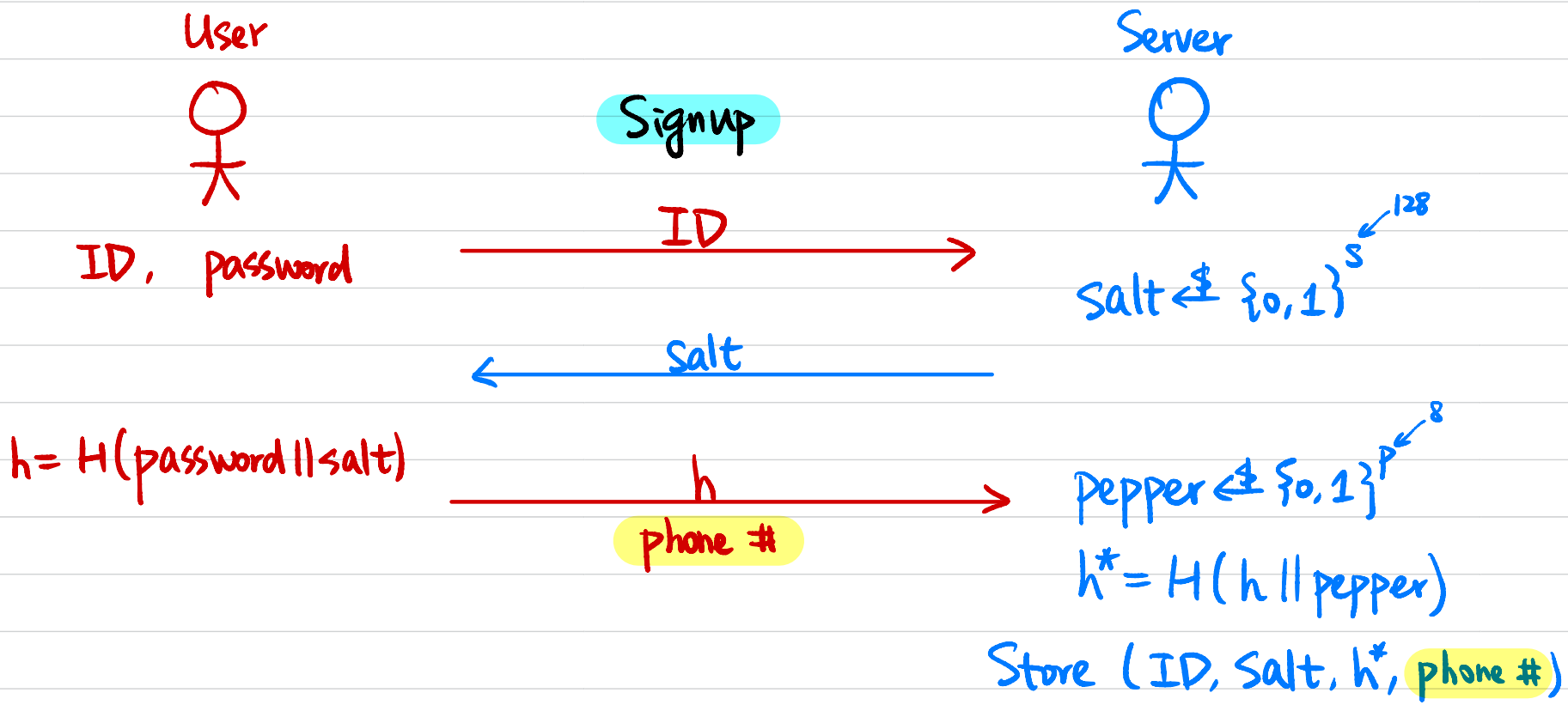
CSCI 1515 Applied Cryptography

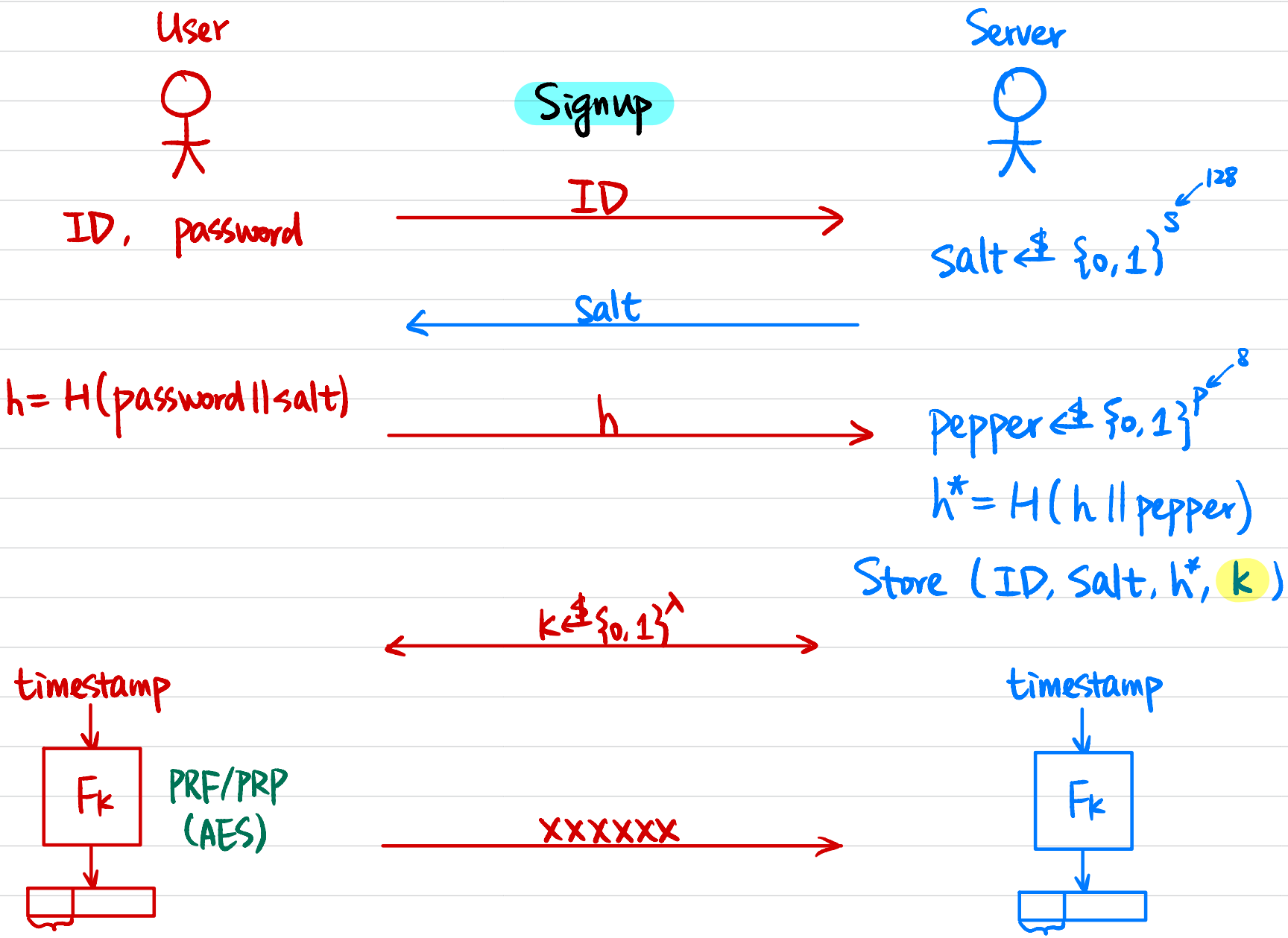
This Lecture:

- Two-Factor Authentication (continued)
- Putting it All Together: Secure Authentication
- Public Key Infrastructure (PKI)
- Case Study: Secure Shell Protocol (SSH)
  Secure Messaging / Group Chats
  Single Sign-On (SSO) Authentication

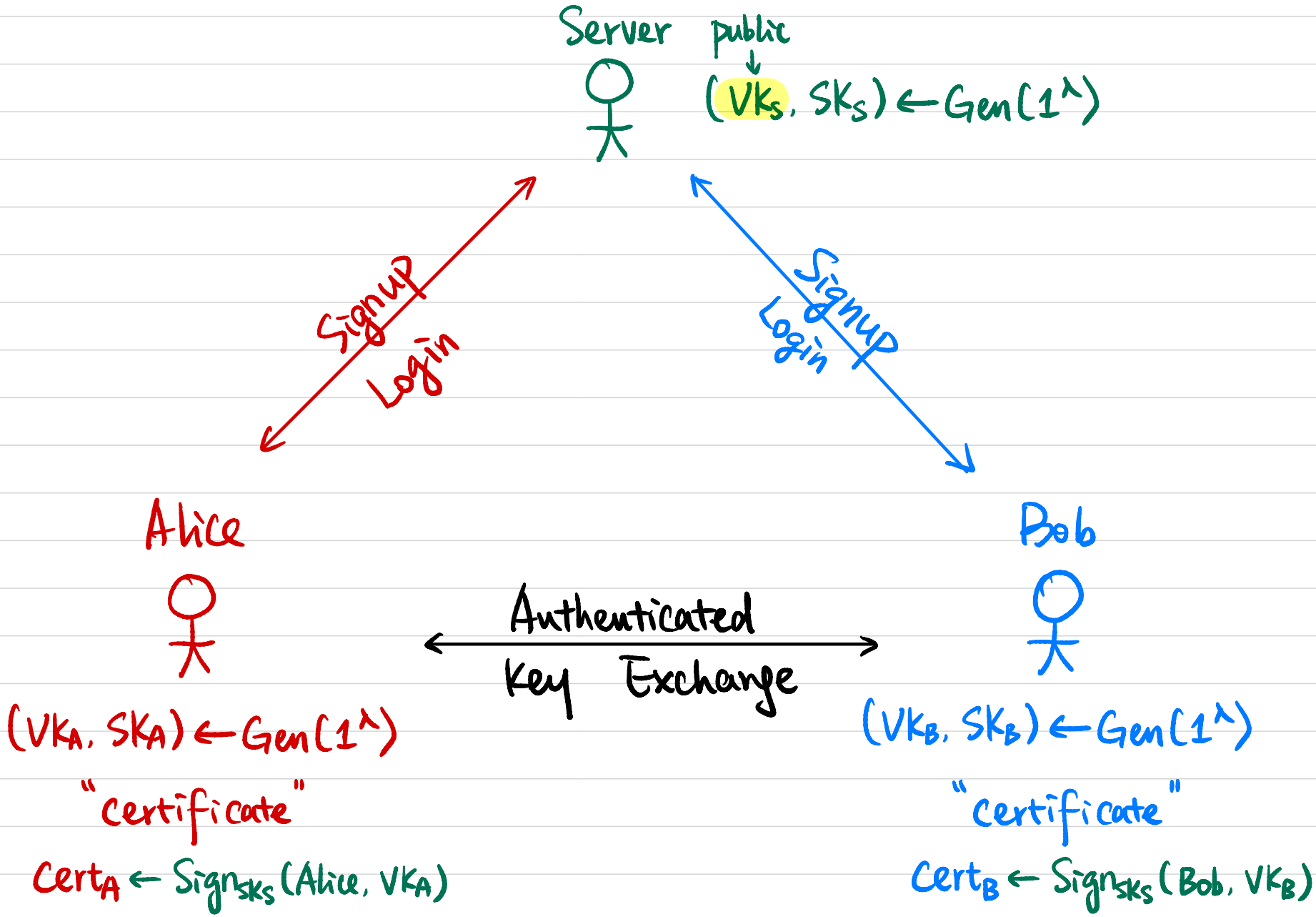
Two-Factor Authentication (2FA) ① SMS



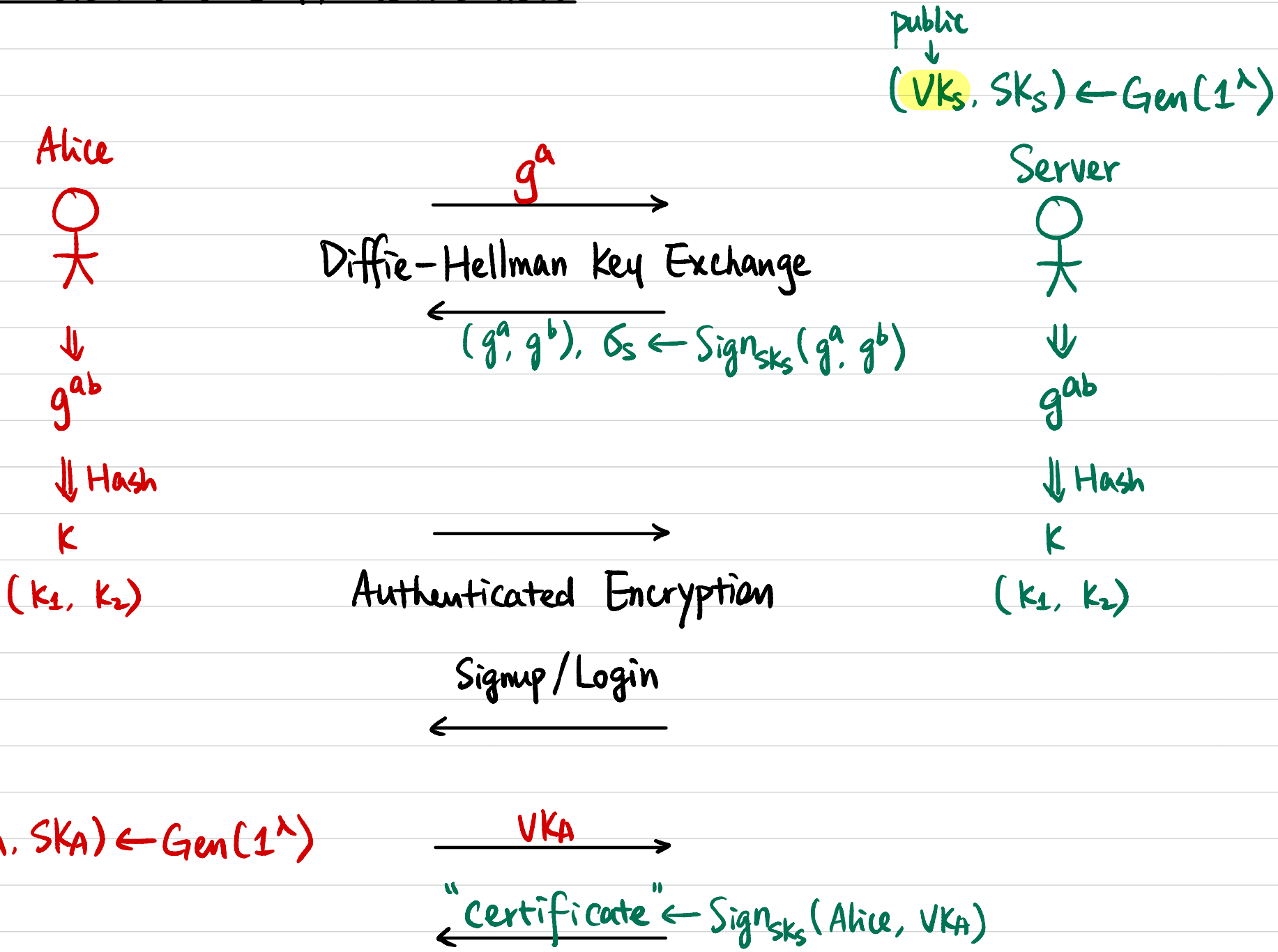
Two-Factor Authentication (2FA) ② app-generated code



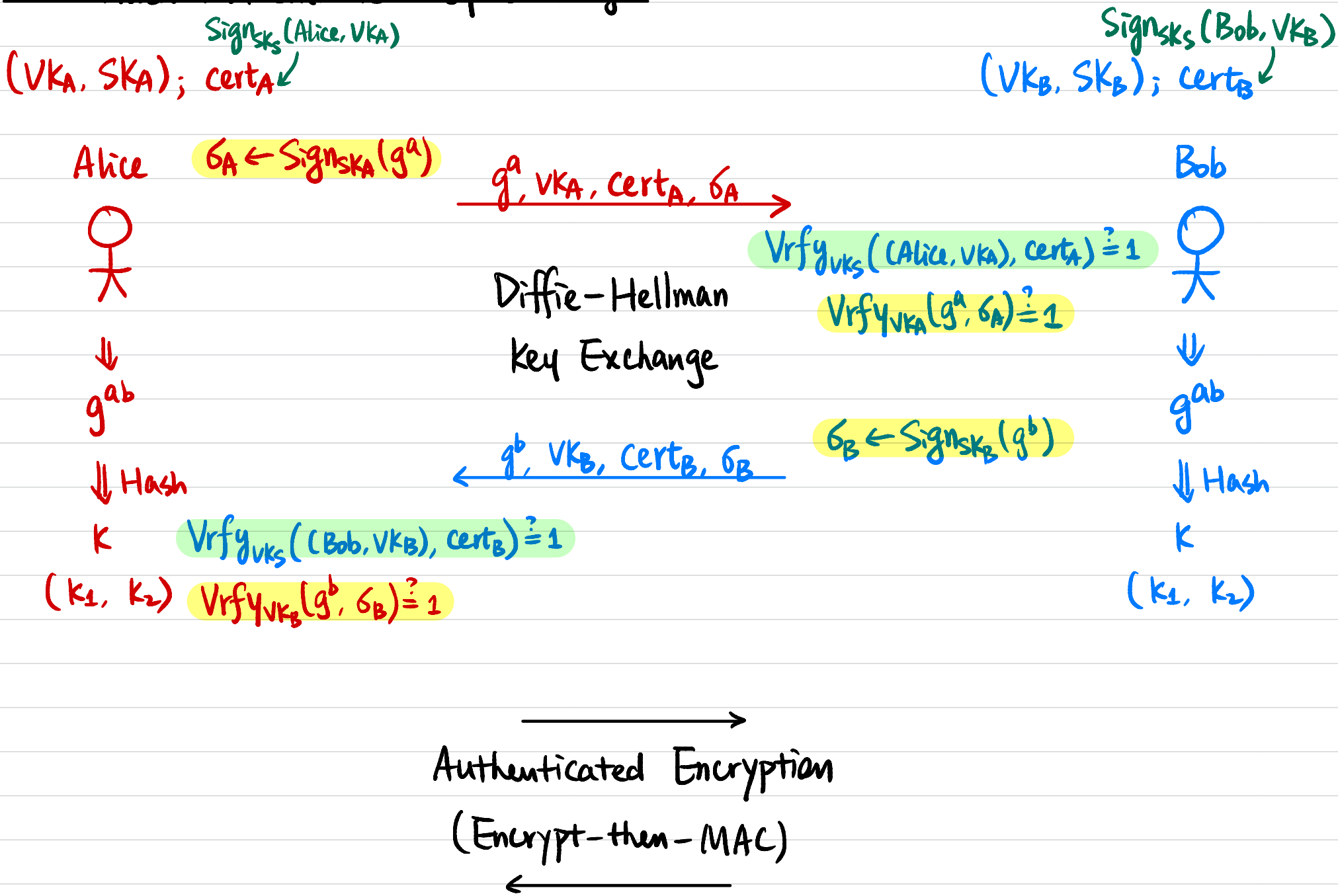
Putting it All Together: Secure Authentication



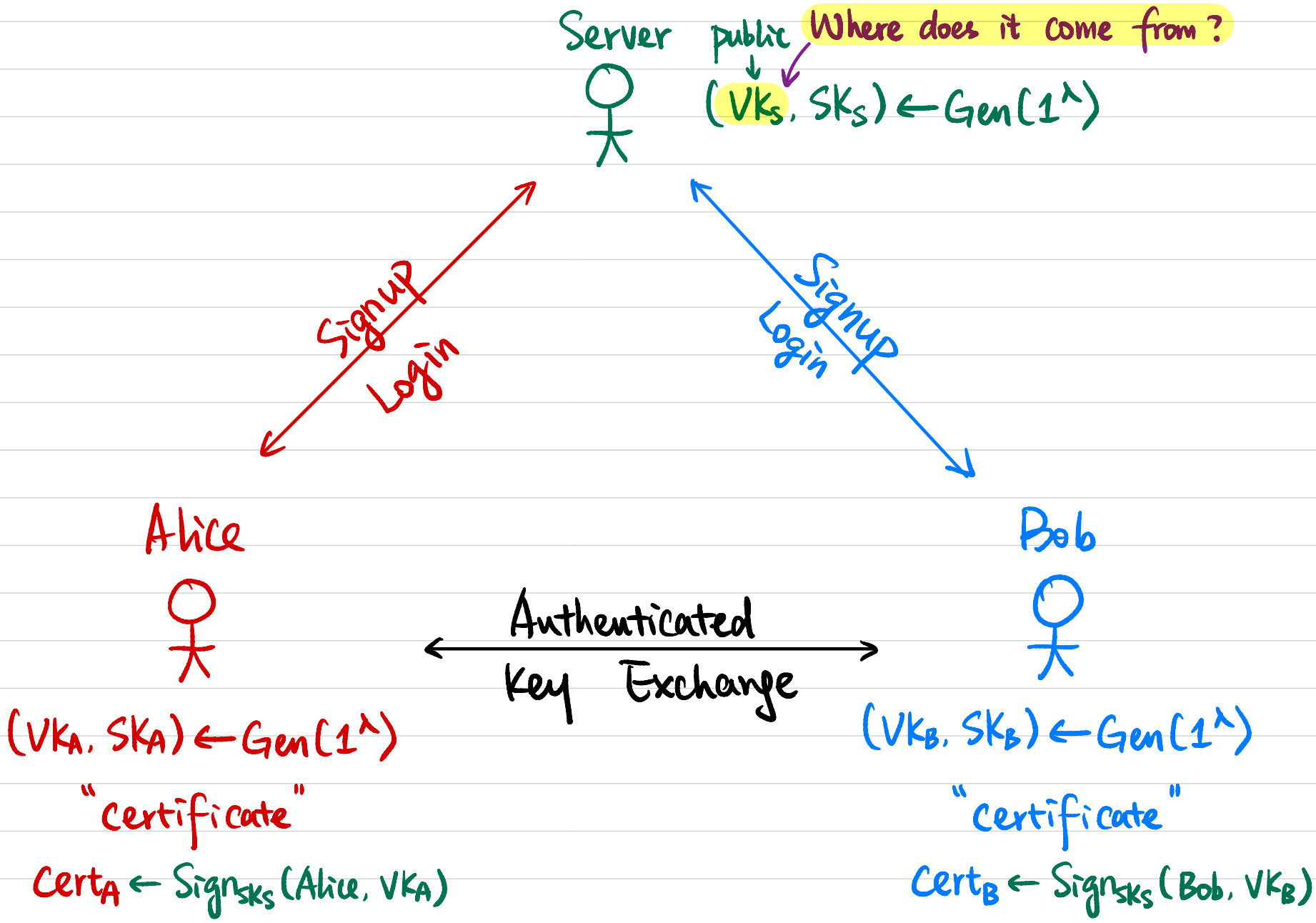
One-Sided Secure Authentication



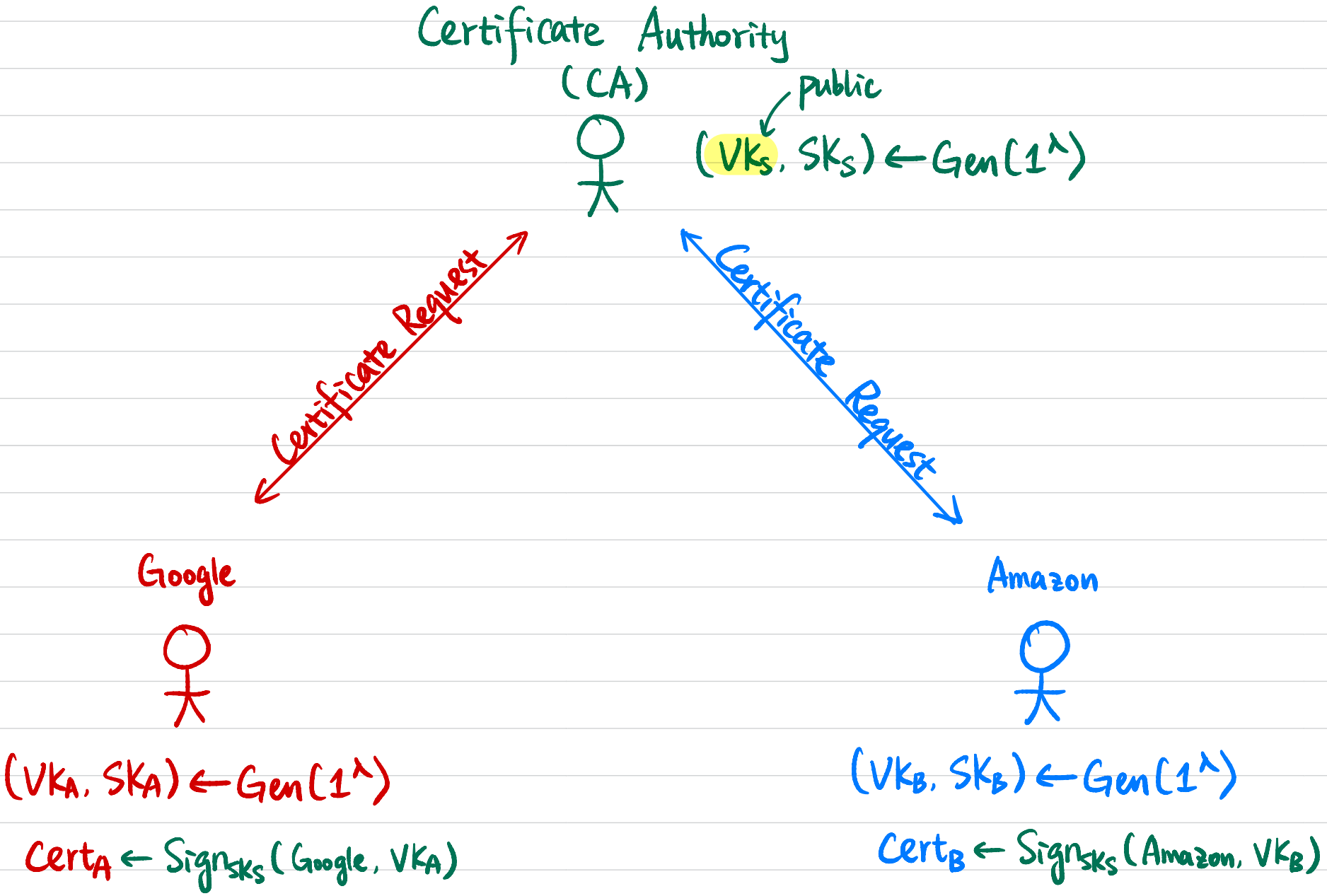
Two-Sided Authenticated Key Exchange



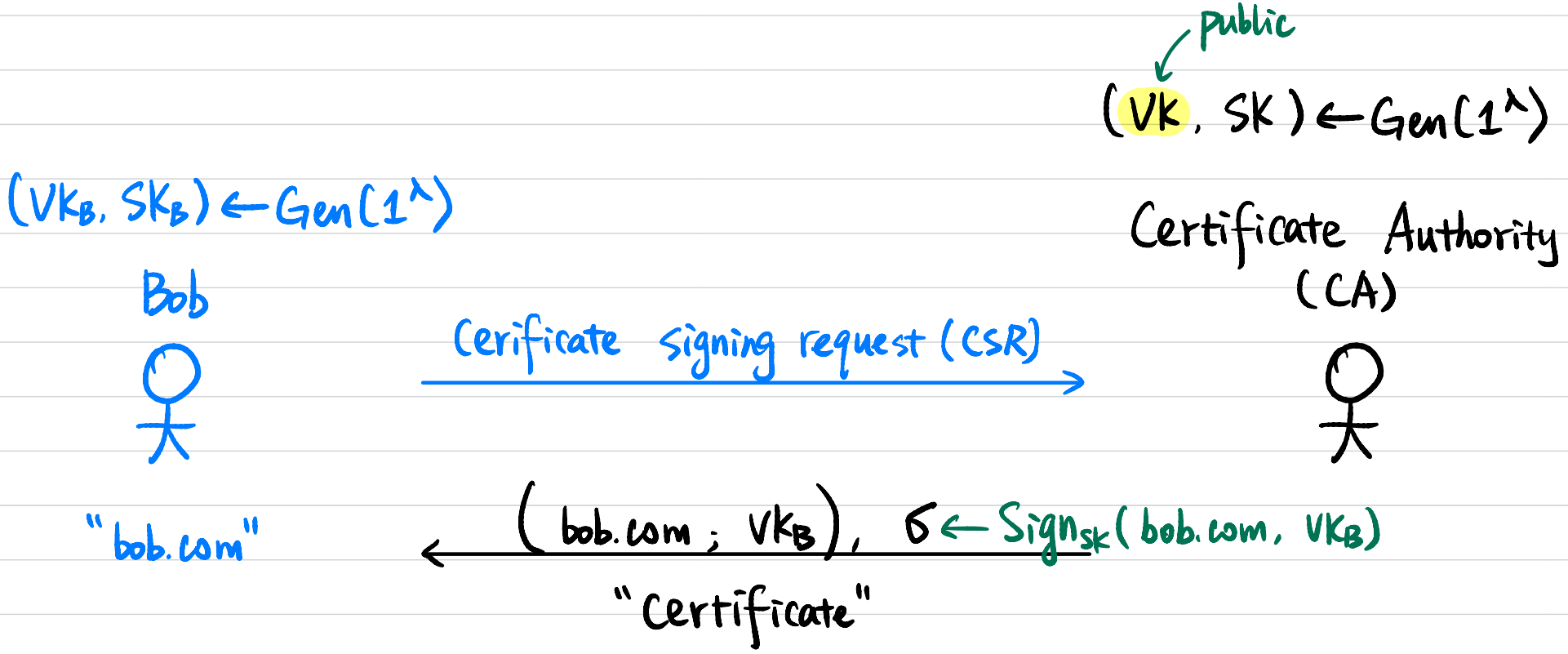
Putting it All Together: Secure Authentication



Public Key Infrastructure (PKI)



Public Key Infrastructure (PKI)



Standard: X.509 certificate

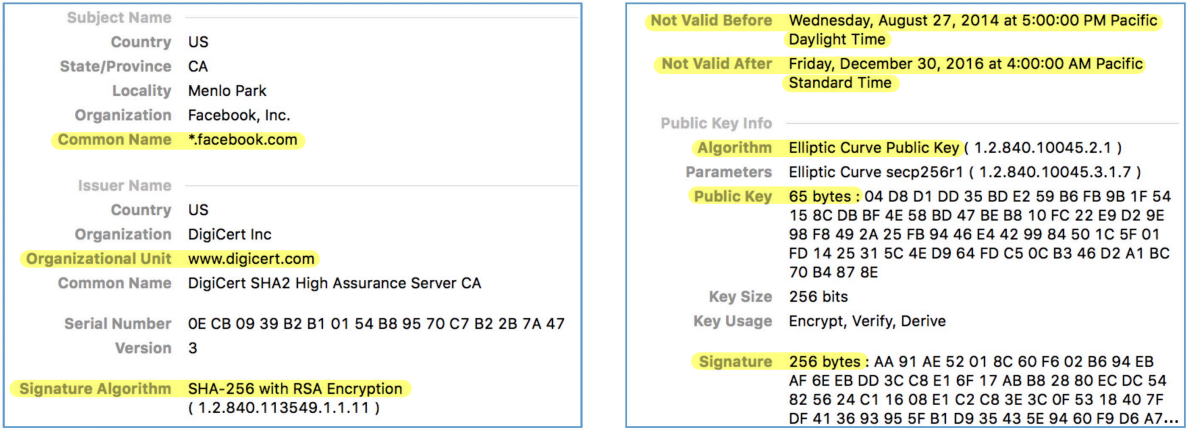
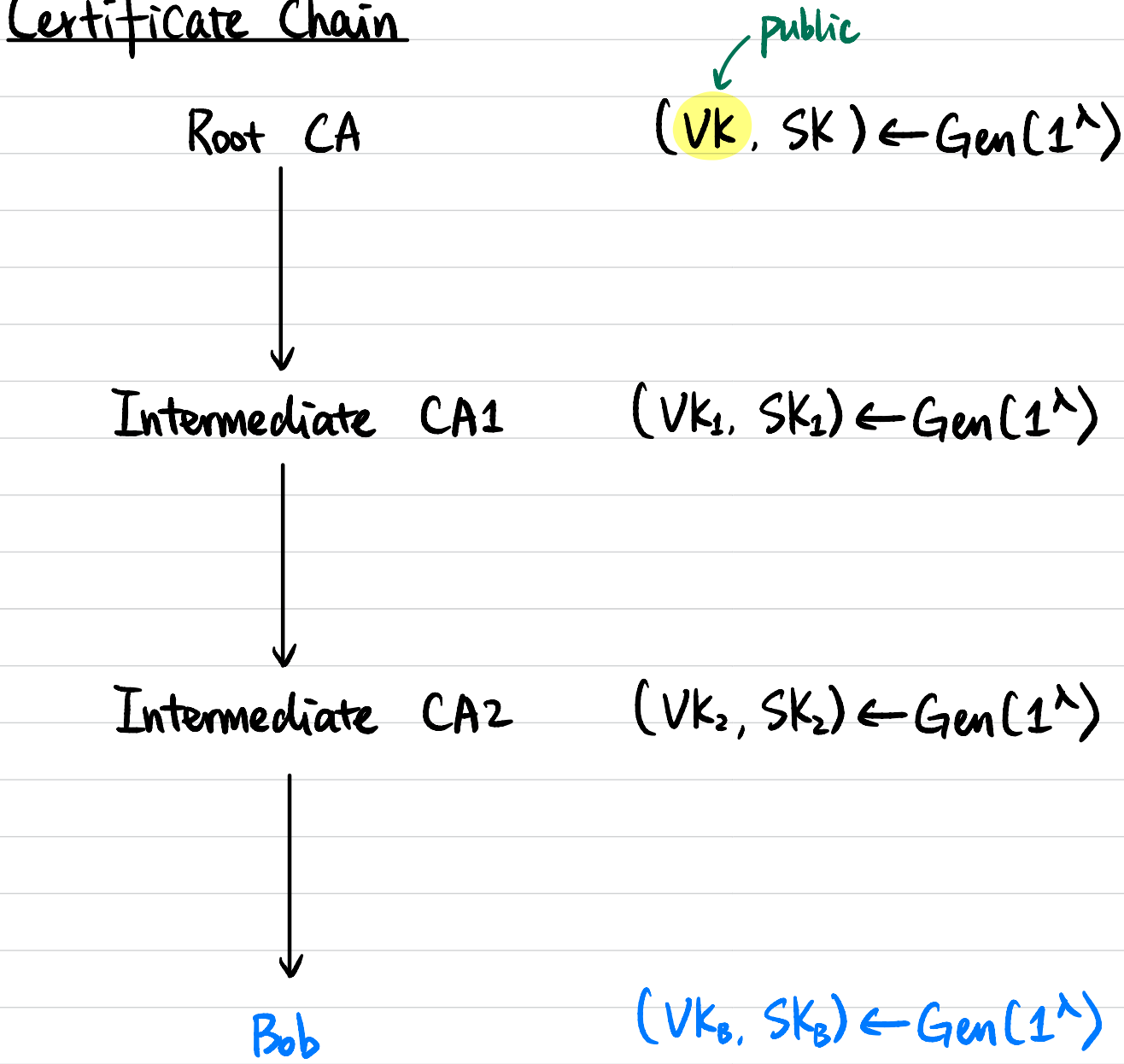


Figure 13.4: An example X.509 certificate

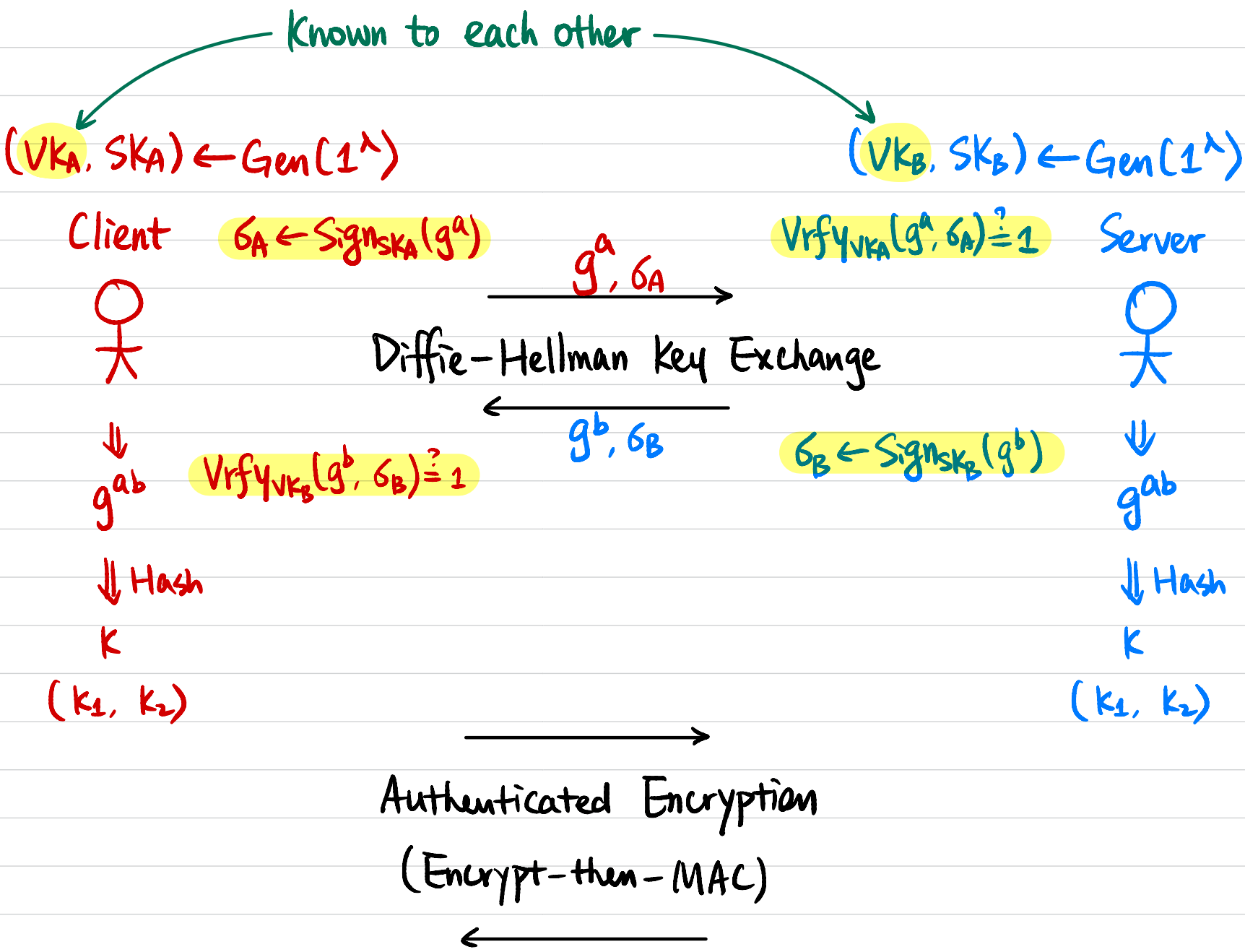
Certificate Chain



Certificate Revocation

- Short-lived certificates
- Certificate revocation lists (CRLs)

Case Study: Secure Shell Protocol (SSH)



Generating a new SSH key

You can generate a new SSH key on your local machine. After you generate the key, you can add the key to your account on GitHub.com to enable authentication for Git operations over SSH.

> **Note:** GitHub improved security by dropping older, insecure key types on March 15, 2022.
>
> As of that date, DSA keys (`ssh-dss`) are no longer supported. You cannot add new DSA keys to your personal account on GitHub.com.
>
> RSA keys (`ssh-rsa`) with a `valid_after` before November 2, 2021 may continue to use any signature algorithm. RSA keys generated after that date must use a SHA-2 signature algorithm. Some older clients may need to be upgraded in order to use SHA-2 signatures.

1. Open Terminal.
2. Paste the text below, substituting in your GitHub email address.

```
$ ssh-keygen -t ed25519 -C "your_email@example.com"
```

> **Note:** If you are using a legacy system that doesn't support the Ed25519 algorithm, use:
>
> ```
> $ ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
> ```

This creates a new SSH key, using the provided email as a label.

```
> Generating public/private ALGORITHM key pair.
```

When you're prompted to "Enter a file in which to save the key", you can press **Enter** to accept the default file location. Please note that if you created SSH keys previously, ssh-keygen may ask you to rewrite another key, in which case we recommend creating a custom-named SSH key. To do so, type the default file location and replace id_ssh_keyname with your custom key name.

```
> Enter a file in which to save the key (/Users/YOU/.ssh/id_ALGORITHM: [Press ent
```

3. At the prompt, type a secure passphrase. For more information, see "Working with SSH key passphrases."

```
> Enter passphrase (empty for no passphrase): [Type a passphrase]
> Enter same passphrase again: [Type passphrase again]
```

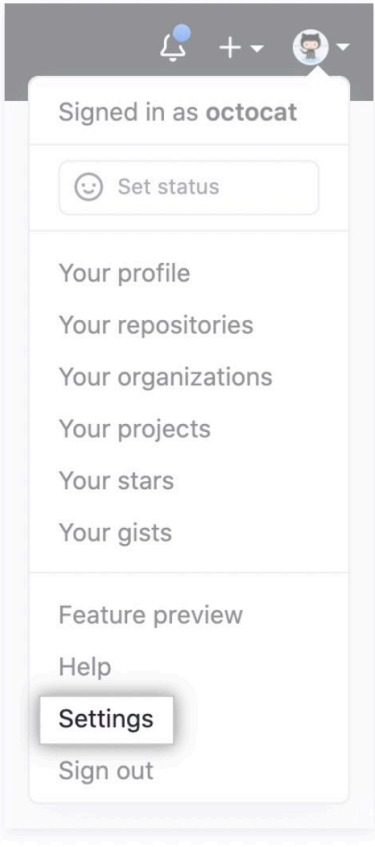
1. Copy the SSH public key to your clipboard.

   If your SSH public key file has a different name than the example code, modify the filename to match your current setup. When copying your key, don't add any newlines or whitespace.

```
$ pbcopy < ~/.ssh/id_ed25519.pub
  # Copies the contents of the id_ed25519.pub file to your clipboard
```

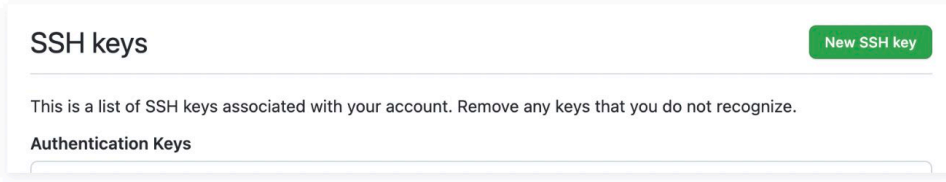
   Tip: If `pbcopy` isn't working, you can locate the hidden `.ssh` folder, open the file in your favorite text editor, and copy it to your clipboard.

2. In the upper-right corner of any page, click your profile photo, then click **Settings**.



3. In the "Access" section of the sidebar, click **SSH and GPG keys**.

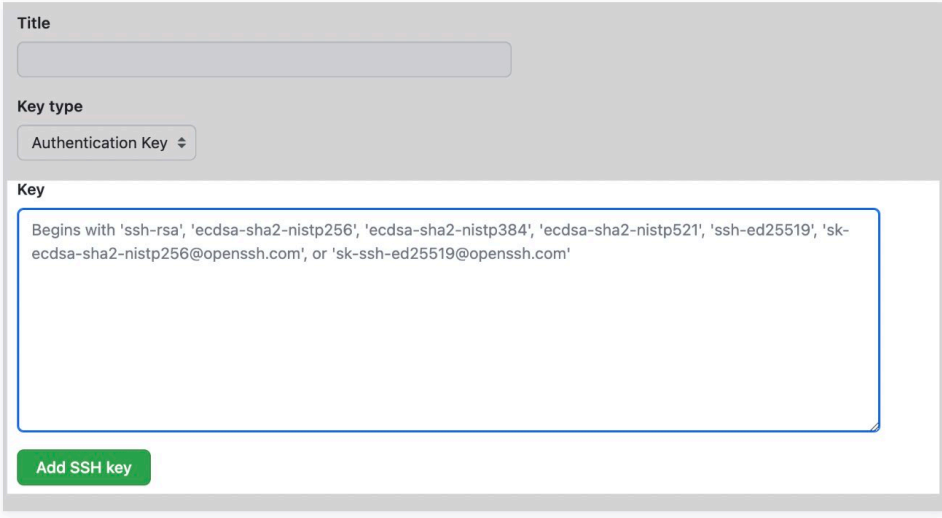
4. Click **New SSH key** or **Add SSH key**.



5. In the "Title" field, add a descriptive label for the new key. For example, if you're using a personal laptop, you might call this key "Personal laptop".

6. Select the type of key, either authentication or signing. For more information about commit signing, see "About commit signature verification."

7. Paste your public key into the "Key" field.



8. Click **Add SSH key**.

9. If prompted, confirm access to your account on GitHub. For more information, see "Sudo mode."

Testing your SSH connection

After you've set up your SSH key and added it to your account on GitHub.com, you can test your connection.

Mac Windows Linux

Before testing your SSH connection, you should have:

- Checked for existing SSH keys
- Generated a new SSH key
- Added a new SSH key to your GitHub account

When you test your connection, you'll need to authenticate this action using your password, which is the SSH key passphrase you created earlier. For more information on working with SSH key passphrases, see "Working with SSH key passphrases".

1. Open Terminal.
2. Enter the following:

```
$ ssh -T git@github.com
# Attempts to ssh to GitHub
```

You may see a warning like this:

```
> The authenticity of host 'github.com (IP ADDRESS)' can't be established.
> RSA key fingerprint is SHA256:nThbg6kXUpJWGl7E1IGOCspRomTxdCARLviKw6E5SY8.
> Are you sure you want to continue connecting (yes/no)?
```

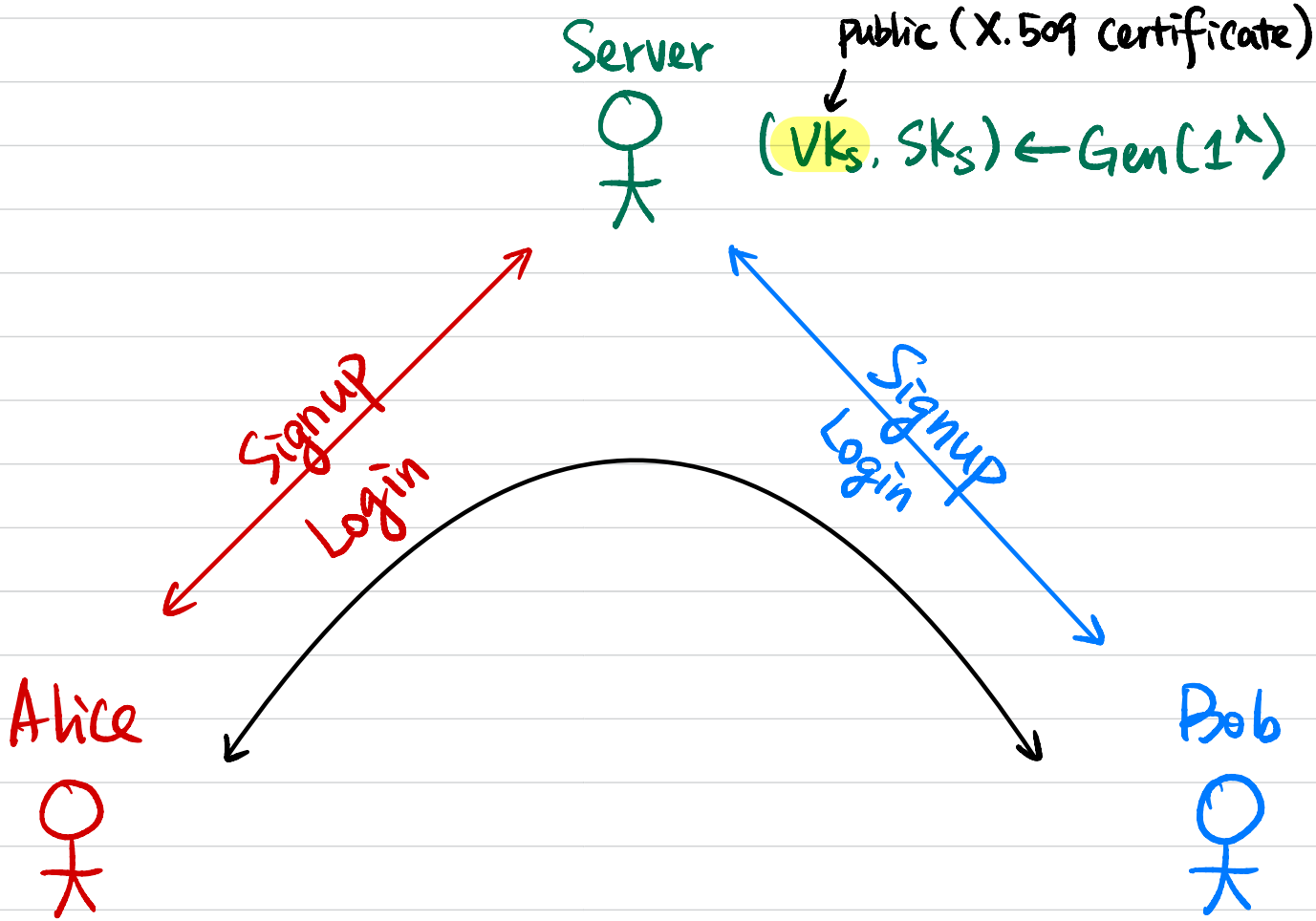
3. Verify that the fingerprint in the message you see matches GitHub's public key fingerprint. If it does, then type `yes`:

```
> Hi USERNAME! You've successfully authenticated, but GitHub does not
> provide shell access.
```

> **Note:** The remote command should exit with code 1.

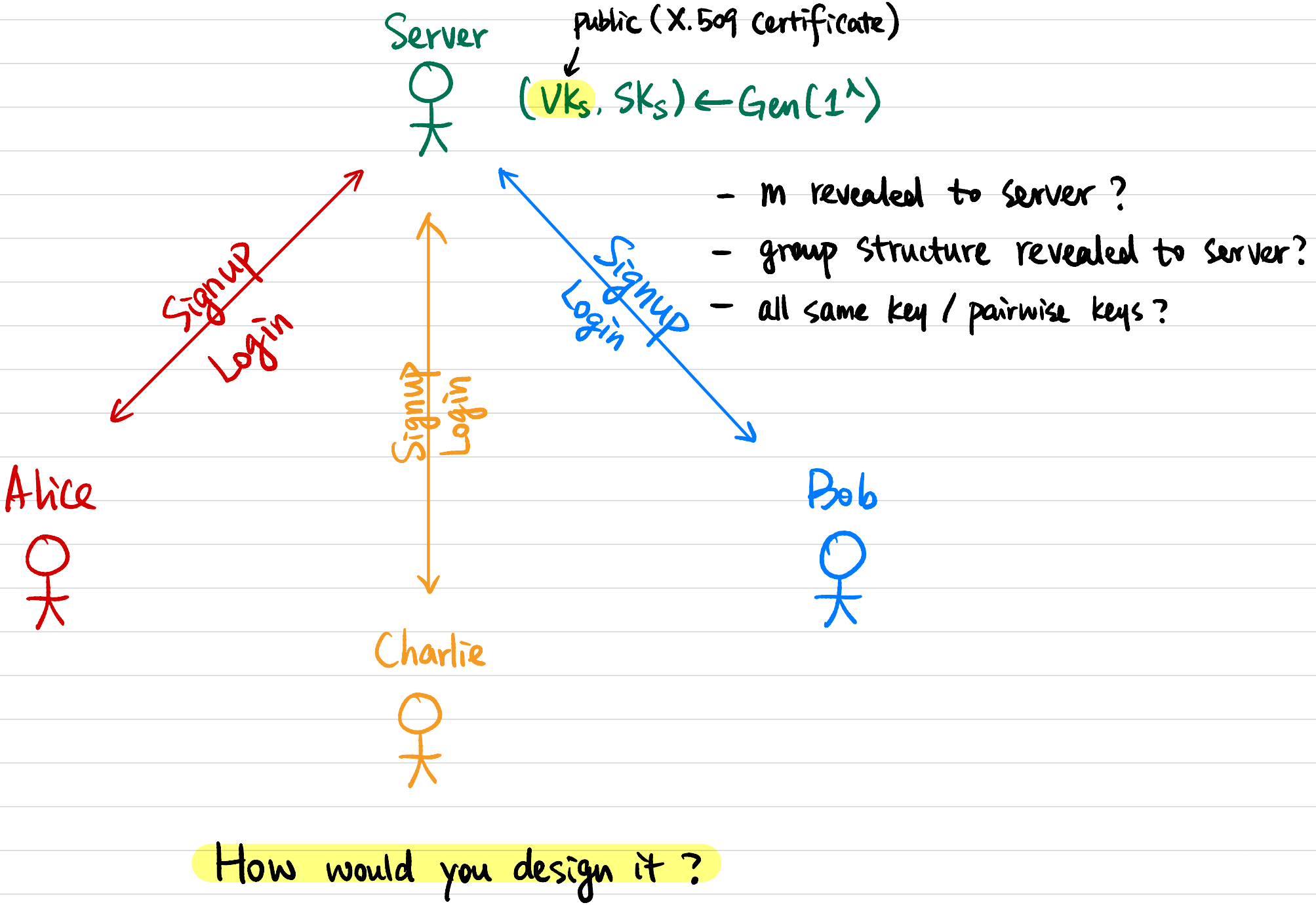
4. Verify that the resulting message contains your username. If you receive a "permission denied" message, see "Error: Permission denied (publickey)".

Case Study: Secure Messaging



How would you design it ?

Group Chat ?



- m revealed to server ?
- group structure revealed to server?
- all same key / pairwise keys ?

How would you design it ?

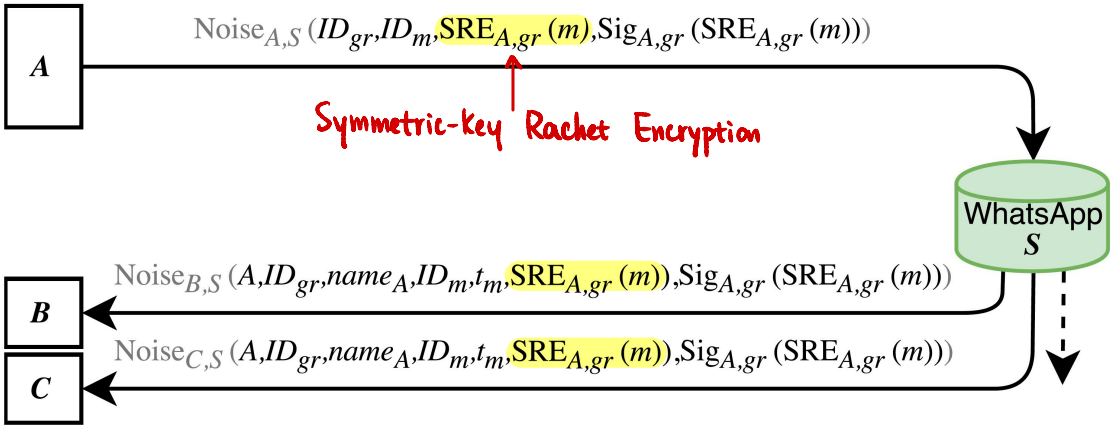


Figure 5. Schematic depiction of traffic, generated for a message m from sender A to receivers B, C in group gr with $\mathcal{G}_{gr} = \{A, B, C\}$ in WhatsApp.

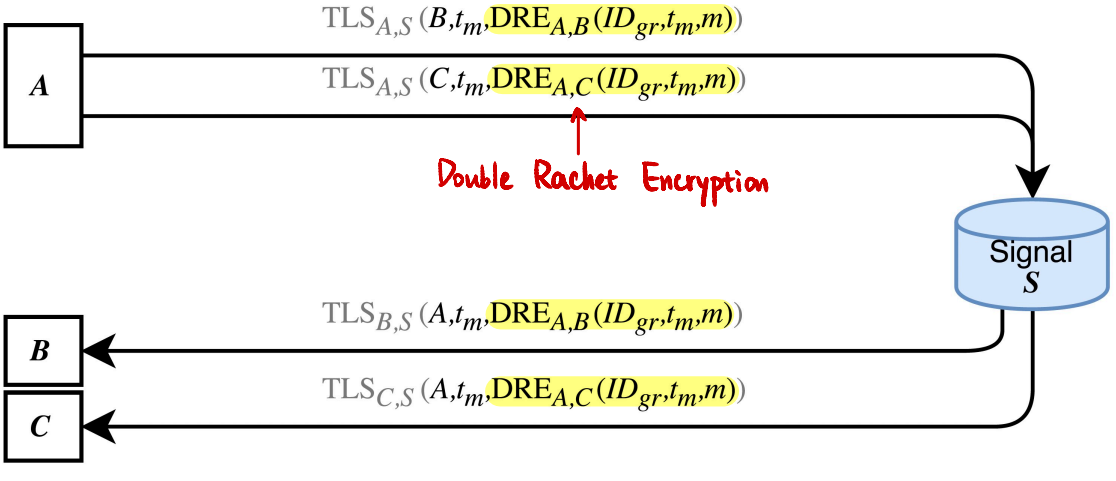
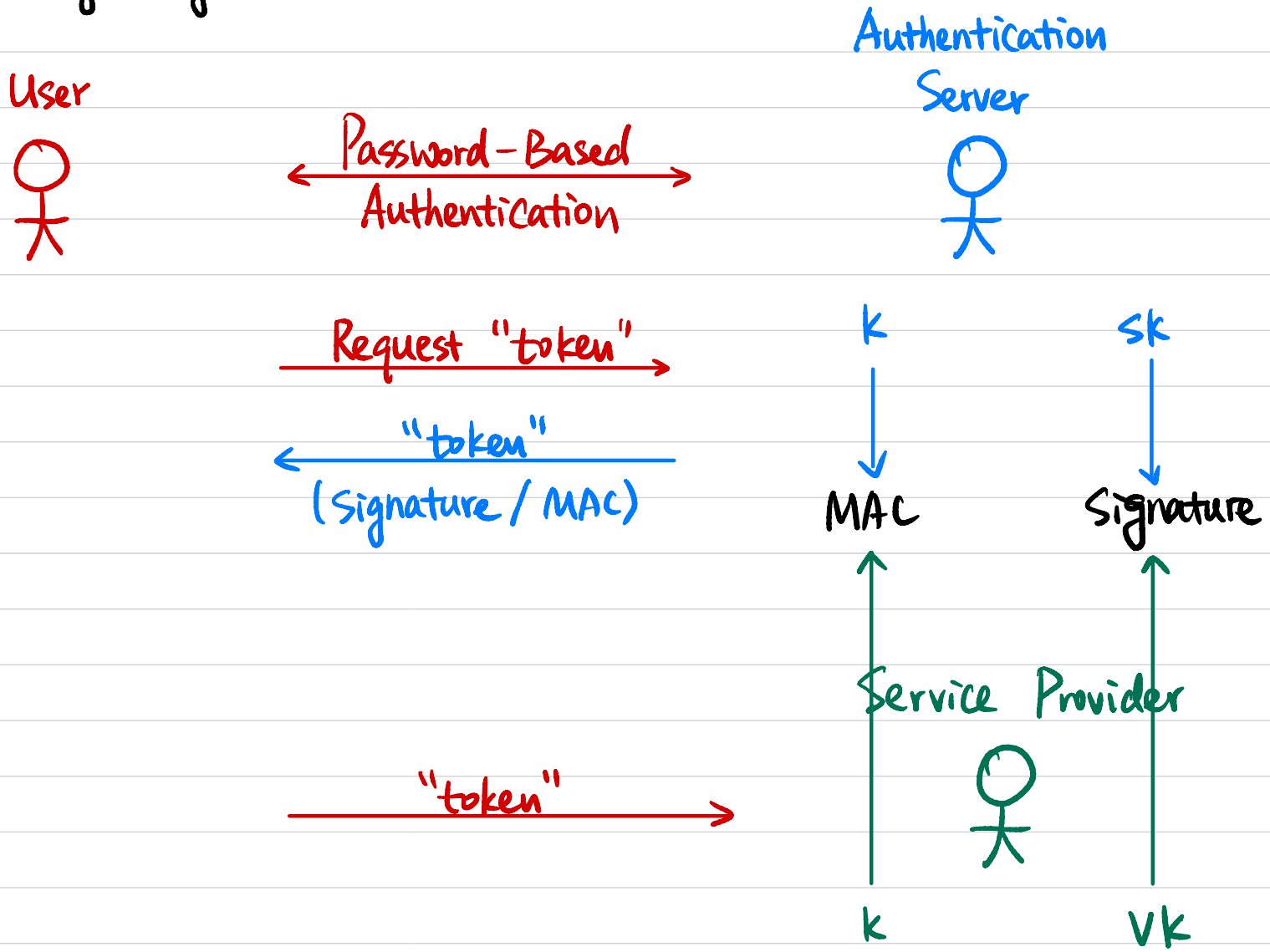


Figure 3. Schematic depiction of Signal's traffic, generated for a message m from sender A to receivers B and C in group gr with $\mathcal{G}_{gr} = \{A, B, C\}$. Transport layer protection is not in the analysis scope (gray).

Case Study: Single Sign-On (SSO) Authentication



- OAuth / OpenID: Sign-in with Google / Apple / Brown / ...
- Kerberos: enterprises