Effective management of SSH access on servers

Ravikumar J. Naik

STCS, TIFR, Mumbai

26/07/2024

Agenda

1. SSH authentication.
2. STCS implementation of centralised key management called Key-Server.
3. STCS use cases.
4. SSH CA implementation for sysadmins to provide an alternative to key-server.

SSH Access:

Unsecured and bad practices

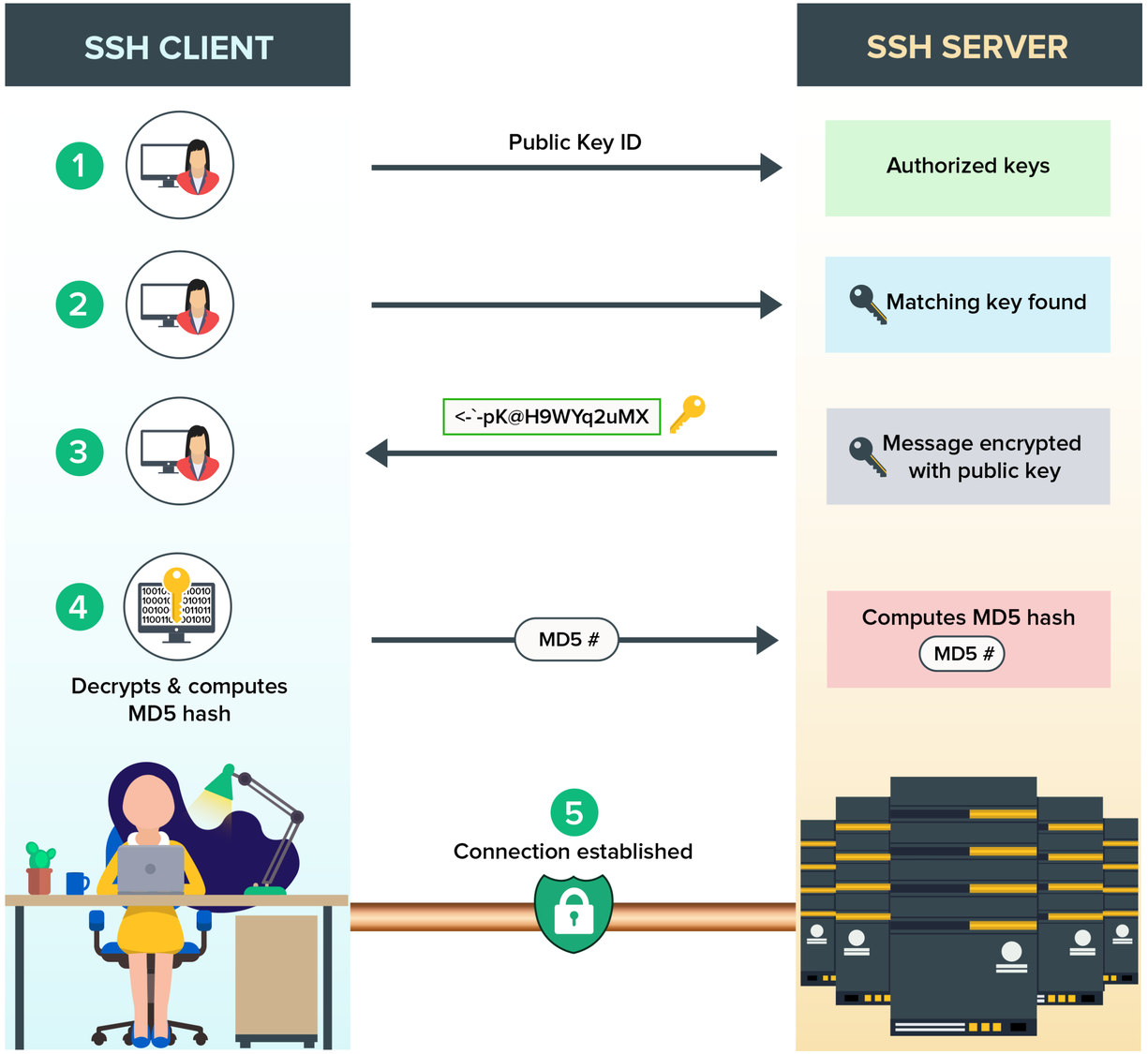
The Secure Shell (SSH) protocol is a method for securely sending commands to a computer over an unsecured network.
SSH uses cryptography to authenticate and encrypt connections between devices.

Security Risks:

1. Weak Passwords:
   - Easily guessable passwords.
   - Vulnerable to brute-force attacks.
2. Password Reuse:
   - Using the same password across multiple servers.
   - Increases the risk of widespread compromise if one server is breached.
3. Poor Password Management:
   - Difficulty in remembering and managing multiple passwords.
   - Reliance on insecure methods of storing passwords.

SSH key authentication

SSH key authentication offers several significant advantages over traditional password-based authentication, enhancing both security and convenience for users and administrators.



SSH key authentication

Advantages

- Enhanced Security:
  - Stronger Authentication: SSH keys provide stronger security than passwords. They use asymmetric encryption, making them much harder to crack than even the strongest passwords.
  - Protection Against Brute-Force Attacks: SSH key pairs are not vulnerable to brute-force attacks, unlike passwords, which can be guessed or cracked through repeated attempts.
- Convenience and Automation:
  - No Need to Remember Passwords: Users don't have to remember or enter passwords each time they log in. This reduces the risk of weak passwords and password reuse.
  - Automation-Friendly: SSH keys are ideal for automated processes, such as scripts and cron jobs, which require SSH access without manual intervention.
- Seamless Multi-Server Access:
  - Single Sign-On Experience: Users can use a single SSH key to access multiple servers without needing to remember different passwords for each one.

SSH key authentication

Security risks and disadvantages

Unauthorized Key Distribution:

Risk: Users might distribute their public keys to multiple systems without proper authorization or tracking, leading to uncontrolled access.

Mitigation: Implement centralized key management and enforce policies for key distribution and authorization.

Key Sprawl:

Risk: Over time, a large number of keys might accumulate in the authorized_keys file, including obsolete or unauthorized keys.

Mitigation: Regularly audit the authorized_keys files on all servers to remove unused or unauthorized keys.

Understanding “AuthorizedKeysCommand” in SSH

What is AuthorizedKeysCommand?

- Purpose: Customizes how SSH obtains public keys for user authentication.
- Usage: Configured in the SSH server configuration file (sshd_config).

Configuration Example

plaintext

```
AuthorizedKeysCommand /usr/local/bin/get-keys %u %h
AuthorizedKeysCommandUser ssh-key-fetcher
```

- **Command**: `/usr/local/bin/get-keys` script retrieves keys.
- **Placeholders**:
  - `%u`: Username
  - `%h`: Home directory
- **User**: Runs as `ssh-key-fetcher` to enhance security.

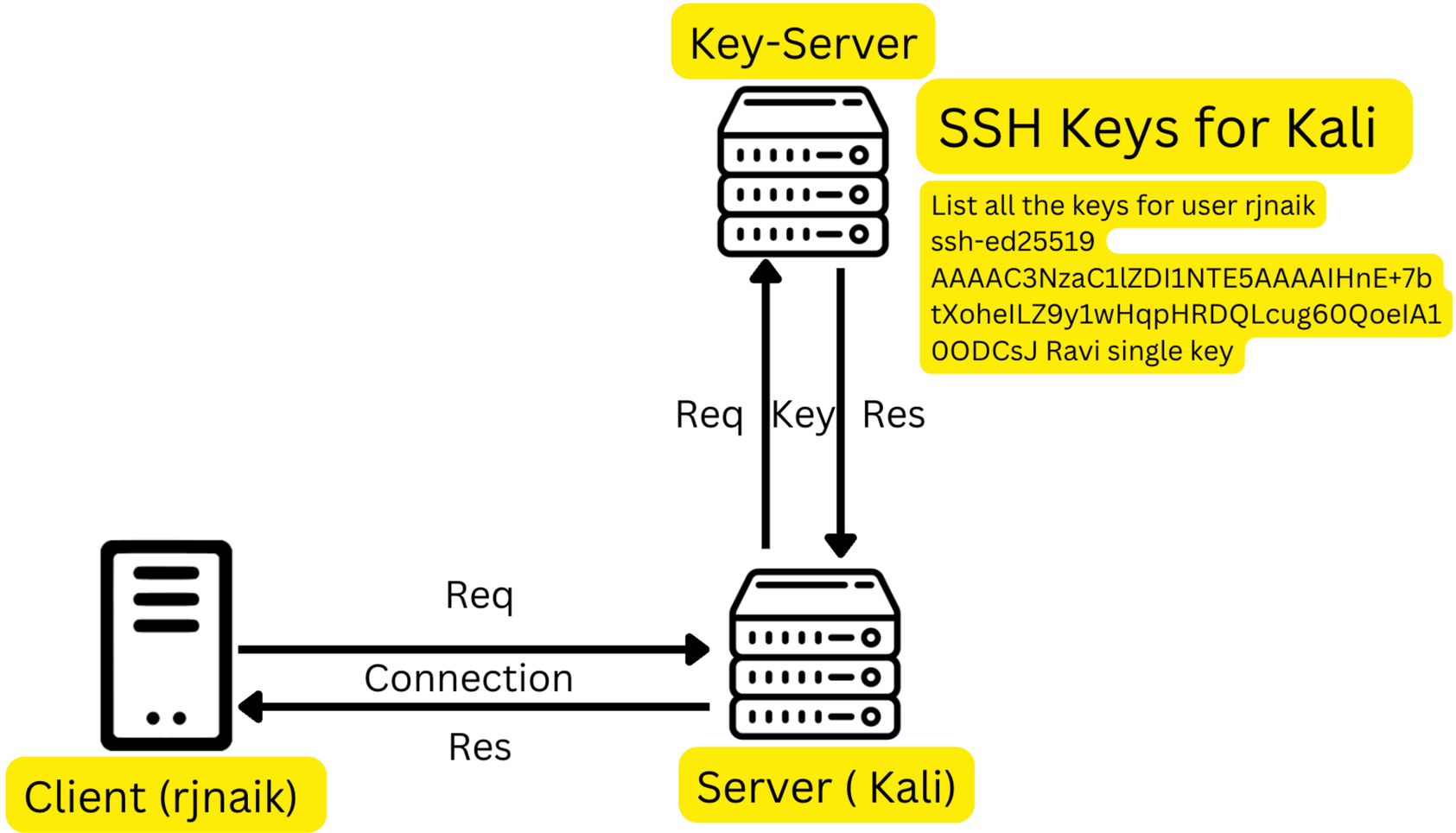
Benefits

- Centralized Key Management: Simplifies key distribution and management.
- Dynamic Key Retrieval: Keys can be fetched from various sources (databases, directories).
- Enhanced Security: Reduces reliance on static key files.

Use Cases

- Enterprise Environments: Central management of SSH keys.
- Dynamic Infrastructure: Fetch keys on-demand for cloud instances.

Key-Server


Key-Server
SSH Keys for Kali
List all the keys for user rjnaik
ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIHnE+7b
tXoheILZ9y1wHqpHRDQLcug60QoeIA1
0ODCsJ Ravi single key
Req
Key
Res
Req
Connection
Res
Client (rjnaik)
Server (Kali)

Key-Server

SSH Server-side setup

SSH configuration on the servers

```
# Fetch keys from the keyserver and do not honour ~user/.ssh/authorized_keys
AuthorizedKeysFile /dev/null
AuthorizedKeysCommand /etc/ssh/fetch_keys.sh %u
AuthorizedKeysCommandUser nobody
```

/etc/ssh/sshd_config

```
#!/bin/bash

# Set these accordingly
# (no trailing slashes for DIRs)
MACHINE_NAME="client"
API_BASE=https://server_url:port/ssh_keys
BA_USERNAME="keyserver_client"
BA_PASSWORD="Password"
CACERT_FILE="/etc/ssh/local-CA.pem"
# (end of changes required)

curl --cacert "${CACERT_FILE}" -u "${BA_USERNAME}:${BA_PASSWORD}" ${API_BASE}/
${MACHINE_NAME}/${1} 2> /dev/null
echo ""
```

fetch_keys.sh

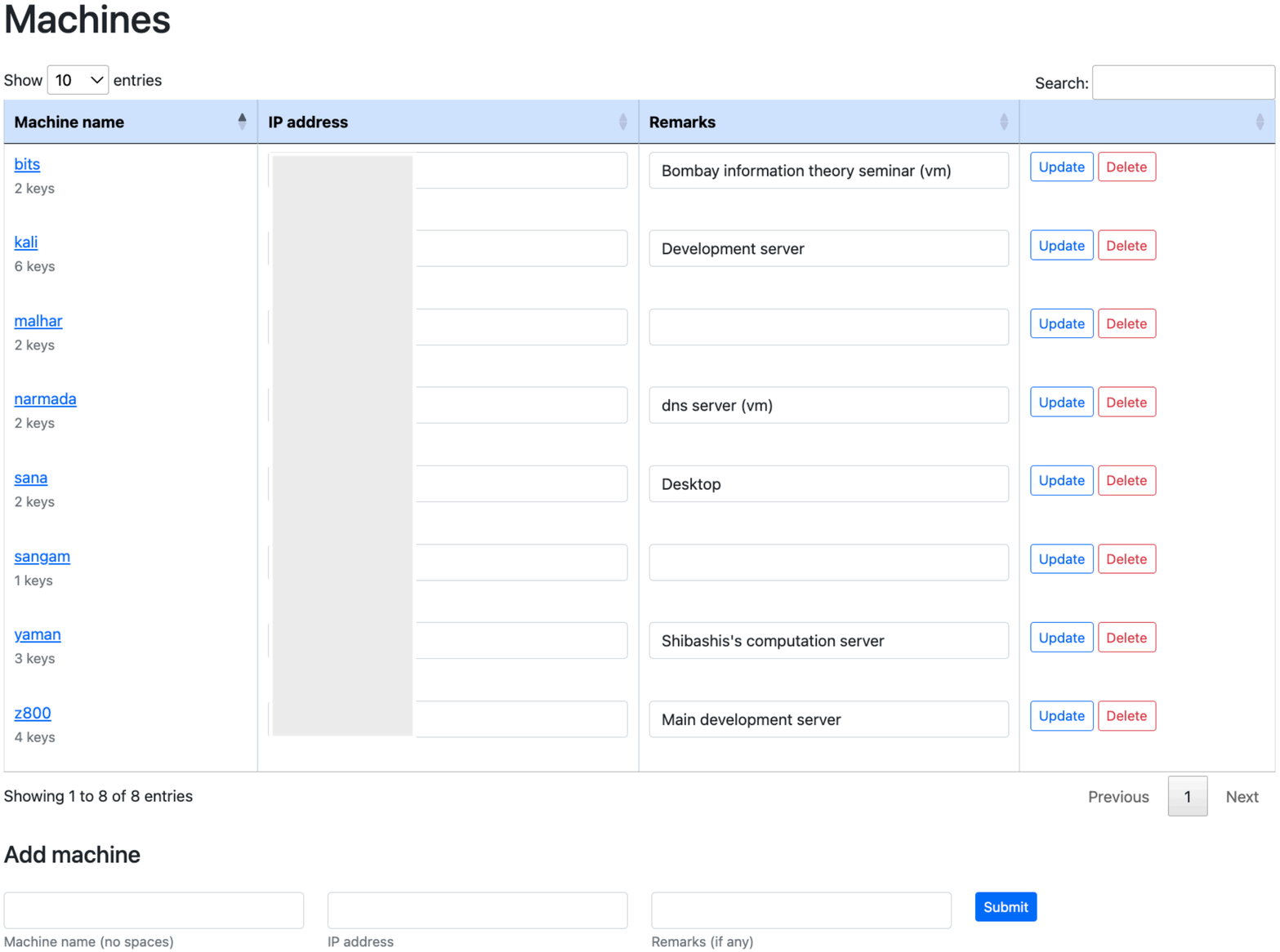
Centralised key management

Key-Server

- We have developed an a web application called Key-Server.
- It maintains a database of ssh keys attached to a specific user and a machine, with a pre- specified expiry date meta data.
- The Key-Server provides a simple api with route to fetch the list of valid ssh public keys to requested user on requesting machine.

Key-Server

Dashboard



Key-Server

Add Key

Back

SSHKeys for kali

Show 10 entries

Search:

Key owner	Username on kali	SSH Key	Created at	Valid until	
Arkadev		(show/hide)	2023-07-12 12:47:11	2026-11-27 00:00:00	Edit Delete
backup user		(show/hide)	2023-06-28 16:55:47	2026-02-26 00:00:00	Edit Delete
Ravikumar Naik		(show/hide)	2023-06-28 16:51:39	2025-07-17 16:51:39	Edit Delete
Sandeep Juneja		(show/hide)	2023-07-26 12:57:53	2026-12-12 12:57:53	Edit Delete
T Kavitha		(show/hide)	2023-08-03 21:14:13	2027-10-03 21:14:13	Edit Delete
Umang Bhaskar		(show/hide)	2023-08-17 22:15:12	2026-12-31 00:00:00	Edit Delete

Showing 1 to 6 of 6 entries

Previous 1 Next

Add new key

Key owner

Local username on kali

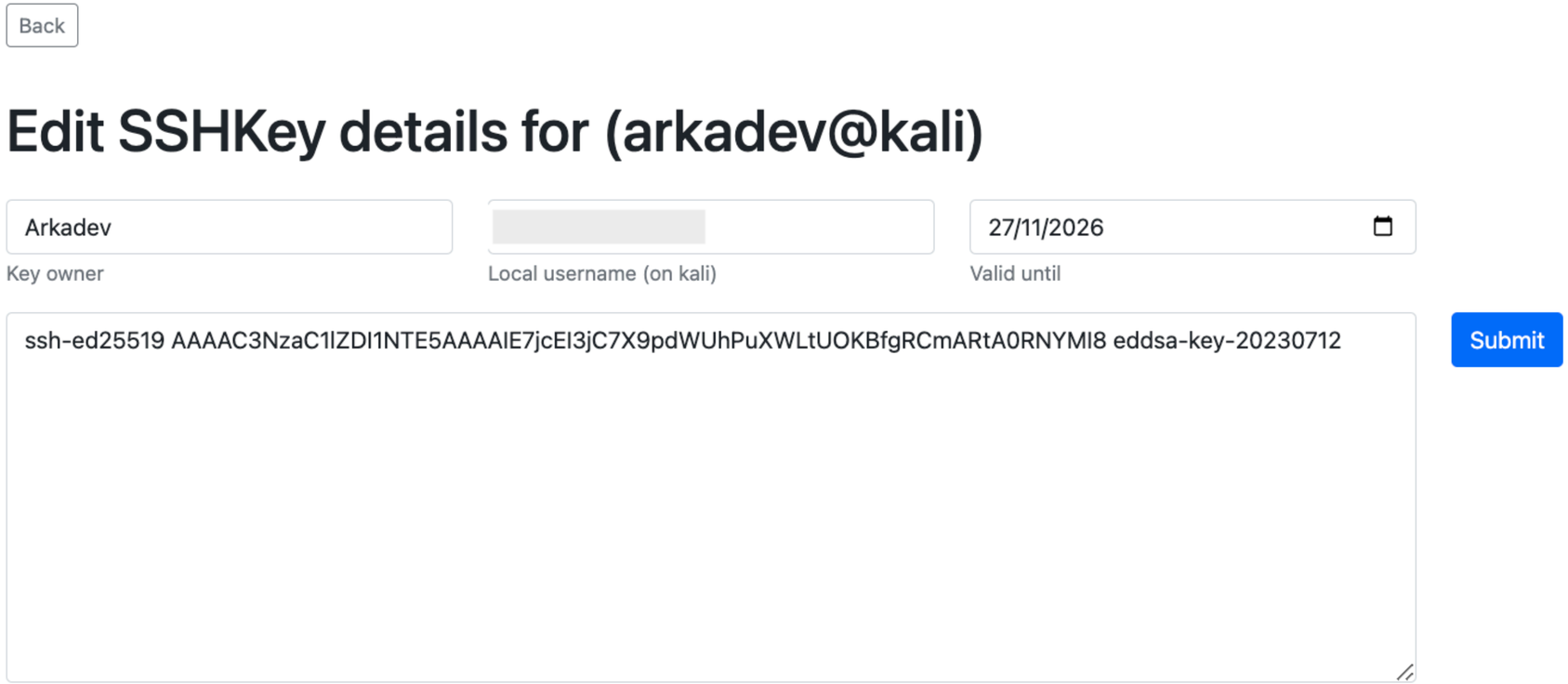
Validity in days

SSH Public Key

Submit

Key-Server

Edit SSH Key details



STCS use cases

- System administrators seamless access to multiple server.
- Automate daily backup process.
  - Our backup server has cron jobs setup to rsync all servers to fetch daily backup.
- STCS faculty, students and interns password-less access to servers.
  - To manage their personal home pages.
  - To access the computational server.

SSH CA

Setup

Create CA key pair using YubiKey.

```
$ ssh-keygen -t ed25519-sk -C "S SSH CA"
-N "" -f s-ssh-ca
```

The above command creates a key pair of type `ed25519-sk` , which requires the YubiKey

SSH configuration on the servers

```
TrustedUserCAKeys /etc/ssh/trusted_CAs
AuthorizedPrincipalsFile /etc/ssh/principals/%u
RevokedKeys /etc/ssh/revoked_keys
```

/etc/ssh/sshd_config

- Create the folder `/etc/ssh/principals`. If `sadmin` is the username that the admin uses to login on the server, then create a file `/etc/ssh/principals/sadmin` with just `administrator` on a line.
- Additionally create a revoked keys file `/etc/ssh/revoked_keys` to revoke some user keys because of a known compromise.

SSH CA

Creating signed SSH keys

```
$ ssh-keygen -s "s-ssh-ca" -I "some_certificate_identifier" -n "administrator" -V +1h user.pub
```

The above command will generate a certificate (user-cert.pub) that has been signed by “s-ssh-ca” CA with the validity of 1 hour.

The file user-cert.pub can then be added to the user’s machine in the same folder that contains the public key user.pub and the private key user (their ~/.ssh/ folder perhaps).

Now can login to the server via the usual command:

```
$ ssh sadmin@server -i ~/.ssh/user
```

Any questions?