An SSH Key Management System: Easing the Pain of Managing Key/User/Account Associations

D Arkhipkin[1], W Betts[2], J Lauret[2] and A Shiryaev[1]

[1] Particle Physics Laboratory, Joint Institute for Nuclear Research, 141 980 Dubna, Moscow Region, Russia
[2] Physics Department, Brookhaven National Laboratory, Upton, NY 11973-5000 USA

E-mail: jlauret@bnl.gov

Abstract. Cyber security requirements for secure access to computing facilities often call for access controls via gatekeepers and the use of two-factor authentication. Using SSH keys to satisfy the two factor authentication requirement has introduced a potentially challenging task of managing the keys and their associations with individual users and user accounts. Approaches for a facility with the simple model of one remote user corresponding to one local user would not work at facilities that require a many-to-many mapping between users and accounts on multiple systems. We will present an SSH key management system we developed, tested and deployed to address the many-to-many dilemma in the environment of the STAR experiment. We will explain its use in an online computing context and explain how it makes possible the management and tracing of group account access spread over many sub-system components (data acquisition, slow controls, trigger, detector instrumentation, etc.) without the use of shared passwords for remote logins.

1. Introduction

Remote user authentication via SSH keys provides a two-factor authentication mechanism as well as potential user convenience compared to the use of passwords. However, in an environment in which users need access to multiple accounts on multiple systems, including shared "group" accounts (such as "operator" or even "root"), managing the association of users with user accounts along with the corresponding distribution of SSH keys can be a very tedious process. We have developed a tool for managing SSH access and key distribution to STAR's [1] online computing environment with a gatekeeper model described below.

The use of our SSH key management tool efficiently satisfies several additional operational and cyber security requirements. For instance, via the SSH key fingerprint, we are able to identify the user logging in to accounts with multiple users. If it is necessary to lock out a user (e.g. if a user's private keys may have been breached, or worse), then an administrator can quickly remove that user's access and within minutes all managed nodes will be updated automatically. From the user's perspective, there is nothing required other than a simple web interface to upload his public SSH key into the system and to request access to managed accounts. While other solutions meet some of these goals (*cf* the OpenSSH LDAP Public Key Patch [2]), our package provides a fuller set of features from the user and administrator interface to the client node key retrieval mechanism and backend server components.

2. Software description

The STAR key management software is divided into several components, which we will discuss as the "server side", "client side" and the users' interface. An overview of the system is shown in Figure 1.

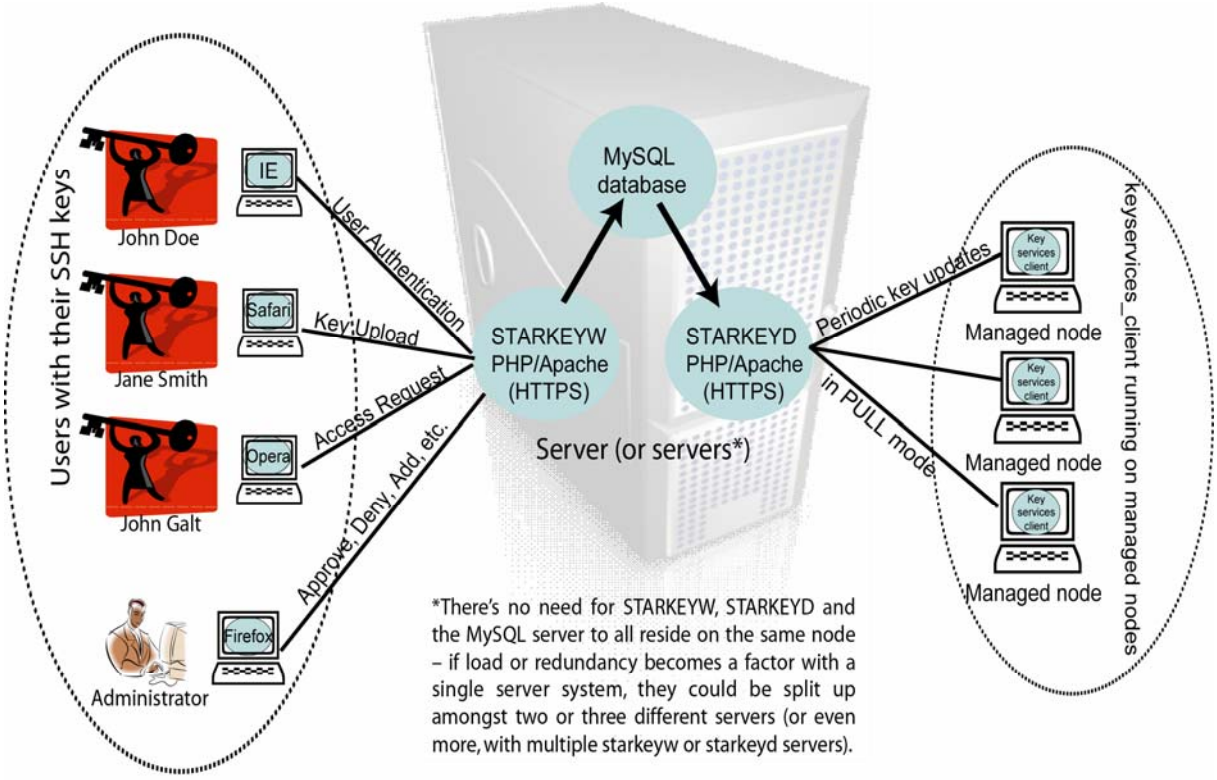

Figure 1: Overview of the STAR SSH key management system. In the middle are the server components that interact with the users (shown on the left) and the client nodes (shown on the right).

2.1 Server side

At the core of the system is a MySQL database containing information about the users, their SSH public keys, the managed nodes (clients), user accounts on the clients and the associations between them all. A database also maintains a log of user and administrator actions. The database schema is shown in Figure 2.

The *STARKEYD* and *STARKEYW* components (both written in PHP) interact with the database on behalf of the client nodes and users, respectively. *STARKEYD* (the 'D' is for "key Distribution") accepts periodic queries from client nodes and provides the clients with updated information from the database, while *STARKEYW* (the 'W' is for "Web interface") provides a web-based interface for users to interact with the system. All interactions with *STARKEYD* and *STARKEYW* are done via HTTPS. For additional security, client nodes can be authenticated via host keys. In STAR's case, *STARKEYW* users are authenticated via a Kerberos system maintained outside of our online facility, which ensures that users are "valid" users within our laboratory's guest and employee appointment system. Other authentication methods are easily substituted in. Administrators or users respectively manage or access the system using a Web interface within their roles.

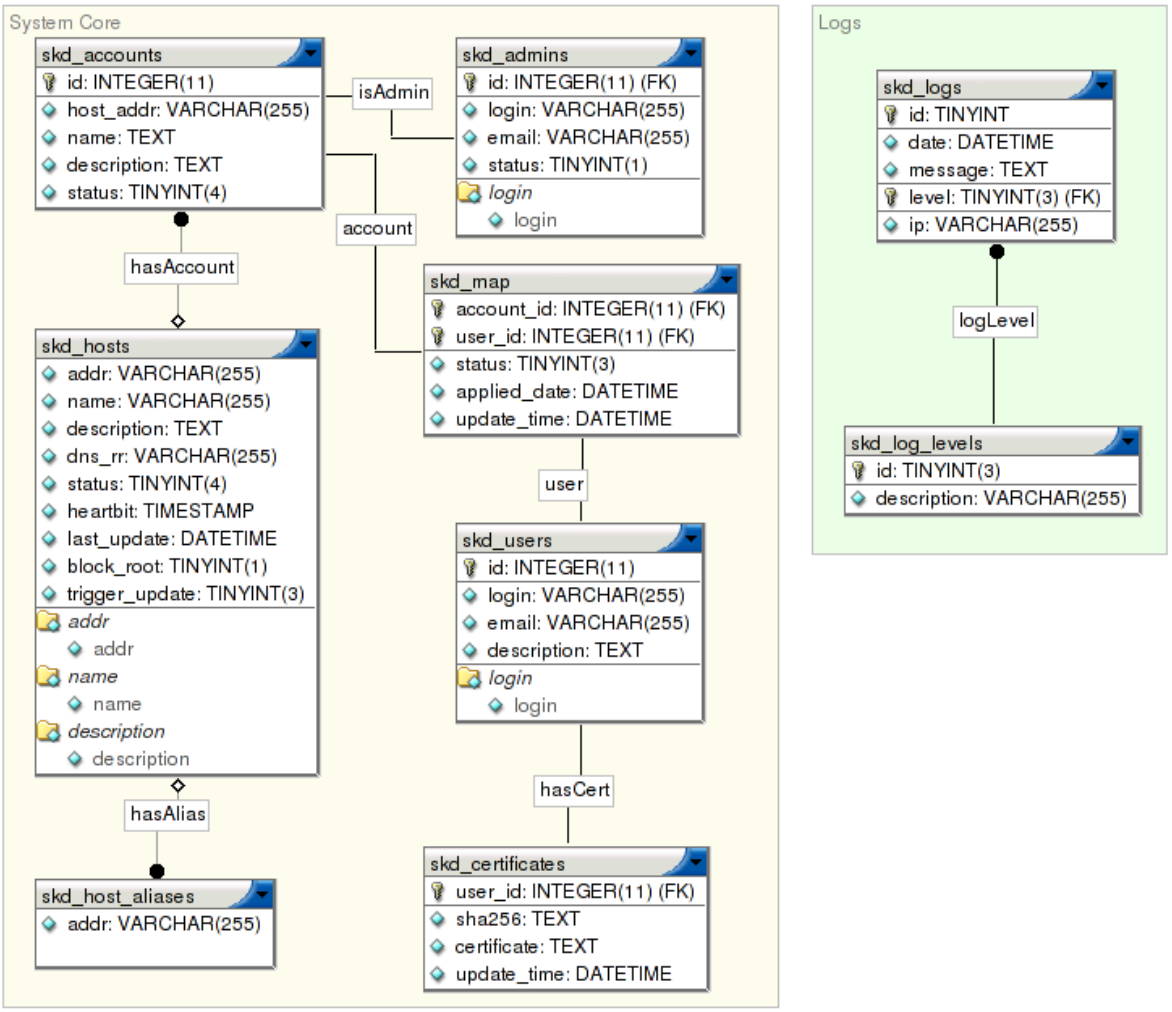

Figure 2: The database schema for the STAR SSH Key Management System

2.2 Client side

On managed nodes, a *keyservices_client* package is installed along with several minor configuration changes. A *STARKEYD* host is specified in the configuration, along with the desired update interval (with a default of 10 minutes). If desired, host keys can be used to authenticate the client and server to each other. The SSH server on a client node is typically modified to deny password authentication and the log level is set to "VERBOSE". With this log level, users' key fingerprints and remote client IP addresses are logged at each login in */var/log/secure*, allowing the trace back of who is actually logging in (and from where) to accounts that have multiple users. The client service also has the feature of removing any keys that have been added outside of the key management system by auser having access to a given node. This feature is important to ensure the system takes full control over the allowed keys, hence, allowing to fulfill our design goals for (a) centralized inventory of access and (b) sole control over authorization and access therefore, bringing confidence and legitimacy of access to the managed accounts.

2.3 User interface

Users (including administrators) have a straightforward web-based interface to the system. The typical sequence of events from a user's point of view goes as follows:

1. A system administrator of a machine named FOO creates a user account named "operator" and, if not done already, installs the *keyservices_client* software on FOO.
2. A key service administrator creates an "operator" account associated with host 'FOO' in the key management administrator interface.
3. John Doe uploads (via the web) his public SSH key (in OpenSSH format).
4. John Doe requests (via the web) that his key be added to operator's *authorized_keys* file on FOO.
5. A key service administrator approves the request, and the *keyservices_client* places the key in *~operator/.ssh/authorized_keys* within a few minutes.
6. John Doe logs in as operator@FOO with his key.

An example of an administrator's interface is shown in Figure 3, which shows the overview of users in the system. From here, an administrator can get more information about individual users, add or remove client hosts and user accounts, check the status of existing client hosts, approve or deny users' pending access requests and examine the logs.

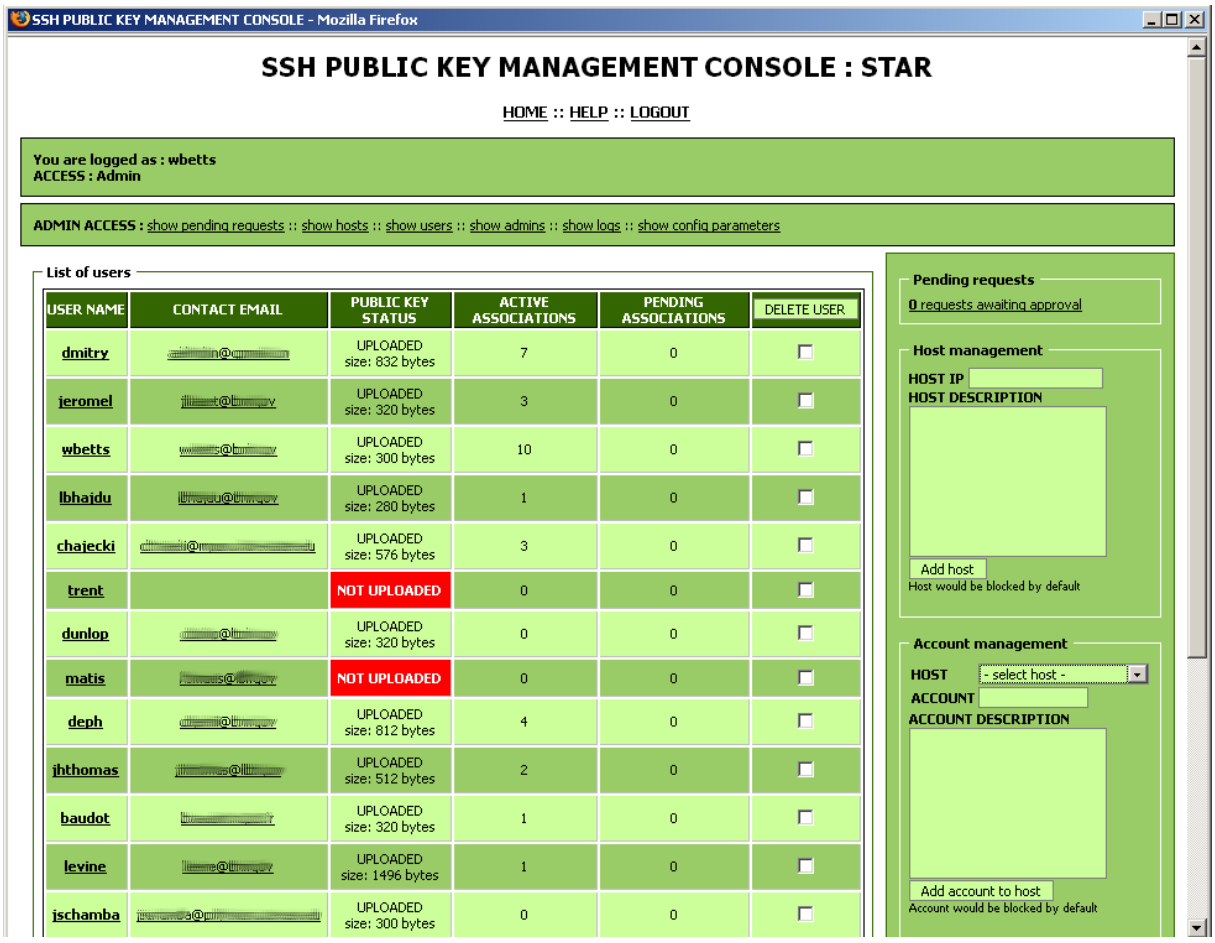

Figure 3. Shown here is one page from an administrator's interface to the STAR key management system, in which the administrator sees an overview of the users on the left and has the opportunity to add hosts and user accounts on the right.

3. STAR's online gatekeeper model

The STAR experiment's online computing resources reside on one firewalled subnet and several internal private networks. A small number of nodes are dual-homed to allow information transfer amongst the various networks. In addition to on-site personnel (such as the shift crews in the STAR Control Room), many sub-system experts require access to STAR systems at all times from off-site. Figure 4 shows the basic access mechanism for a user sitting outside of the STAR experimental area. Assume Jane Smith has uploaded her key to the STAR key management system, requested an account on the STAR gatekeeper as well as access to two "group" accounts, the "root" account on node1 and the "operator" account on node2. If she is not at Brookhaven National Lab (site of the STAR experiment), she will first have to log in to one of the lab's SSH gateway nodes, via whatever means they support (SSH keys and CryptoCards being the current favourites). In this way, STAR has essentially effortless assurance that her employee or guest status is still valid at the time of her log in. Once logged in to a node at the lab, she can proceed to log in to one of the handful of STAR gatekeeper nodes inside the STAR experiment's firewalled subnet using her SSH key and from there proceed to access additional clients of the key management system. With SSH key forwarding enabled, she can make multiple hops without any need for passwords or passphrases.

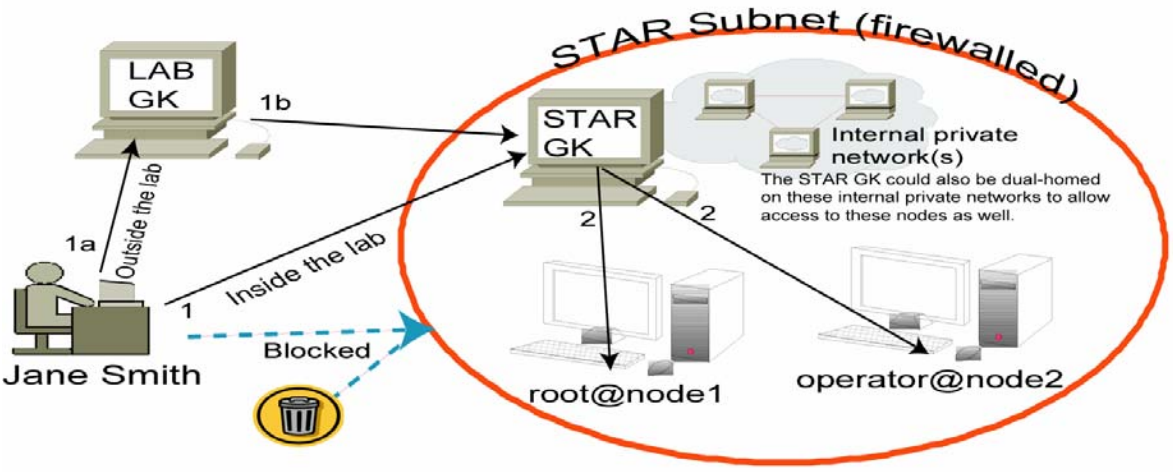

Figure 4. Pictorial representation of the STAR experiment access control for sub-system experts trying to log in remotely. The STAR Gatekeeper nodes are client nodes in the STAR key management system. Therefore, Jane Smith can log into a gatekeeper from a remote location using her SSH key (via path (1) or (1a+1b) depending on her location). After the first login with her SSH key, key forwarding allows additional logins to approved accounts on additional key management client nodes on the STAR experimental subnet (2), such as root@node1 and operator@node2.

4. Technical requirements and additional information

We have produced *keyservices_client* RPMs for Redhat Enterprise Linux and Scientific Linux versions 3 and 4 with 32-bit and 64-bit systems. The principal requirement for the client is xmlrpc-c greater than 1.06.08. For the server components, MySQL 4.1 or higher is required for the standard database structure, and the *STARKEYD* and *STARKEYW* components require PHP 4.4.4 or 5.1.6 (or higher), including the *mhash* and *mcrypt* modules. We have additional information and ongoing documentation of features and possible improvements available online [3].

5. Conclusion

We have developed a system for distributing approved SSH keys to Linux clients that satisfies several common requirements for multi-user, multi-node facilities. The use of SSH keys for remote access naturally satisfies the need for two-factor authentication. Our database of users, client nodes, user accounts and the associations among them is easily updated via our web interface. Therefore, the

burden of managing the distribution and deletion of keys to heterogeneous nodes is greatly reduced to only a few mouse clicks. As changes are made, we have an automatically updated and easily understood record of who has access to what. Via the SSH key fingerprints, we also gain a record of who is actually logging into accounts, which is especially important for accounts that have multiple users.

Acknowledgements
This work was supported by the Office of Nuclear Physics within the U.S. Department of Energy's Office of Science and the Russian Ministry of Science and Technology.

References
[1] "STAR - Solenoidal Tracker at RHIC," http://www.star.bnl.gov
[2] "OpenSSH LPK," http://dev.inversepath.com/trac/openssh-lpk
[3] "STAR SSH Key Management," http://drupal.star.bnl.gov/STAR/comp/onl/tools/sshkeymngt