Practical Privacy-Preserving Authentication for SSH
or, My New Favorite PSI Application

Lawrence Roy
Stanislav Lyakhov
Yeongjin Jang
Mike Rosulek

Oregon State University

To appear at USENIX 2022
ia.cr/2022/740
short presentation for TPMPC 2022
SSH client

should I authenticate
with pub key 6c6c6568...?  

no

SSH server

no

problems:
I server can finger client, (based on its keys for other servers)
I client can tell whether server recognizes some pk, (without knowing its sk)
I other issues too
SSH client

should I authenticate with pub key 6c6c6568...?  

no

should I authenticate with pub key 73616664...?  

no

SSH server

problems: I server canngerprint client, (based on its keys for other servers) I client can tell whether server recognizes some pk, (without knowing its sk) I other issues too
SSH client  SSH server

should I authenticate with pub key 6c6c6568...?  no

should I authenticate with pub key 73616664...?  no  no

...  

yes
SSH client: should I authenticate with pub key 6c6c6568...?  
  no

SSH server: should I authenticate with pub key 73616664...?  
  no

problems:

- server can fingerprint client, (based on its keys for other servers)

- client can tell whether server recognizes some $pk$, (without knowing its $sk$)

- other issues too
our result:

new pk authentication protocol for SSH:

- server learns one bit: whether client knows sk for some authorized pk

- client learns which of its keys are authorized (but only keys for which it knows the sk)

- supports existing SSH keys (RSA, EC-DSA, EdDSA)

- fast!
techniques

Flavor of private set intersection, where:

- parties sets contain $pks$
- client can’t include $pk$ in their set without knowing correct $sk$
- server learns only whether intersection is nonempty
- supports mixture of RSA, ECDH keys

Implementation integrated into OpenSSH client+server for 100 server keys + 20 client keys:
- time comm ECDH keys only 27ms 12kb
- RSA keys only 300ms 54kb

see ia.cr/2022/740 for more details
techniques

Flavor of private set intersection, where:

- parties sets contain pk's
- client can’t include pk in their set without knowing correct sk
- server learns only whether intersection is nonempty
- supports mixture of RSA, ECDH keys

implementation

integrated into OpenSSH client+server for 100 server keys + 20 client keys:

<table>
<thead>
<tr>
<th></th>
<th>time</th>
<th>comm</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECDH keys only</td>
<td>27ms</td>
<td>12kb</td>
</tr>
<tr>
<td>RSA keys only</td>
<td>300ms</td>
<td>54kb</td>
</tr>
</tbody>
</table>
**techniques**

Flavor of **private set intersection**, where:

- parties sets contain *pk*
- client can’t include *pk* in their set without knowing correct sk
- server learns only whether intersection is nonempty
- supports *mixture* of RSA, ECDH keys

**implementation**

integrated into OpenSSH client+server for 100 server keys + 20 client keys:

<table>
<thead>
<tr>
<th></th>
<th>time</th>
<th>comm</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECDH keys only</td>
<td>27ms</td>
<td>12kb</td>
</tr>
<tr>
<td>RSA keys only</td>
<td>300ms</td>
<td>54kb</td>
</tr>
</tbody>
</table>

see ia.cr/2022/740 for more details