Managing Teams and Keys with Keybase

Max Krohn (https://keybase.io/max)
"just scp it from my machine"
• Federated management was better than what we have today but was never good enough.

• Managed apps in the cloud: maybe that ship has sailed

• But at the very least, can we decentralize trust and key management?
Basic Requirements

• Multi-device support
  • Get new phone for Christmas, enter username and password, and get instant access to all history
• Namable teams with mutable membership
• Authenticated invitation of new members
Threat Model

- Bad guys own any server infrastructure
- Bad guys can recover locked device
GCHQ has proposal to surveill encrypted messaging and phone calls. The idea is to use weaknesses in the “identity system” to create a surveillance backdoor. This is a bad idea for so many reasons. Thread. 1/

Principles for a More Informed Exceptional Access...
GCHQ officials outline how to enable the majority of the necessary lawful access without undermining the values we all hold dear.

lawfareblog.com
Slack Warns Investors It's a Target for Nation-State Hacking

As Slack prepares to go public, the company is warning potential investors that it's a target for malicious attacks from “sophisticated organized crime, nation-state, and nation-state supported actors,” according to an SEC filing published today.
Security Goals

• Future messages are not available to a revoked device

• Forward-secrecy is opt-in per-message and can be layered on top (outside scope)
Insufficient Solutions
One Private Key, Encrypted With Password

• Keybase v0

• Most “browser crypto”

• What’s compelling about this idea?

• What’s wrong with this idea?
Keybase’s Approach

• Users think about “devices” not “keys”

• Each device in a user’s cloud is equally powerful. Why?
  • We’ve all lost phones, laptops, slips of paper
  • The more devices, the less likely you are to lose your data
  • And you’re most likely to discard your oldest device

• Reuse this abstraction for teams:
  • Devices are to Users as Users are to Teams
How Apps Work

• Every team has a random shared symmetric key that rotates when:
  • Users are removed from the team
  • Or any team member revokes a device

• All updates to the chat channel (or git repo or file system) are:
  • Encrypted for current shared team symmetric key
  • Done, right?
Encryption, Take 2

- Authenticated encryption in all cases
- Signed by the user that made the update
  - To prevent Alice from putting words into Bob’s mouth
akalin 1:31 PM
https://thenextweb.com/dd/2019/01/05/github-now-gives-free-users-unlimited-private-repositories/

GitHub now gives free users unlimited private repositories
Finally!

zanderz 1:41 PM
@mlsteele @chrisnojima @max The WW/ War of the Worlds mashup I mentioned: https://vimeo.com/107454954

Vimeo

Great martian war
Archive recreation taken from The Great Martian War documentary by impossible factual for History Canada. Directed by Mike Slee. VFX/Animation
Director: Christian...
Lecture Outline

• How devices sign statements to constitute a user
• How users sign statements to constitute a team
• Lessons Learned
How to Define a User
Account Creation

• Picks a new username $n$

• Rolls a new Ed25519 Signing Key Pair $(s,S)$

• Rolls a new Curve25519 DH Key Pair $(d,D)$

• Rolls a new “per-user-key” Curve25519 DH Key Pair $(u,U)$

• Signs $D$ with $s$

• Encrypts $u$ for $D$

• Crucially, $s$ and $d$ never leave the device; encryption of $u$ does

• Posts 3 sigchain links to the Keybase Merkle Tree under $n$
Link 1: 
Alice=S, $\sigma_s(Alice=S)$

Link 2: 
$\sigma_s(D, \text{Hash}(\text{link1}))$

Link 3: 
$\sigma_s(U, \text{Hash}(\text{link2}))$
New Device Addition

- New Ed25519 Key: (s', S')
- New Curve25519 Key: (d', D')
- Signs S with s' and S' with s
- Signs D' with s' as before
- Encrypts u for D'
- Posts 2 new sigchain links
Link 4: $\sigma_s(S', \sigma_{s'}(S), \text{Hash}(\text{link3}))$

Link 5: $\sigma_{s'}(D', \text{Hash}(\text{link4}))$
Ready to provision using *home mac mini - meuse.*

On *home computer*, go to Devices ➔ Add new ➔ New phone.

feito secret instead
Revoking a Device

- Sign a statement to revoke S and D from lost/stolen/retired device

- Rotate per-user-key to \((u', U')\), and re-encrypts \(u'\) for all non-revoked devices

- Encrypts \(u'\) for \(u\)

- Lesson from experience: Watch out for hidden \(O(n^2)\) behavior!
Link 6: \( \sigma_{s'}(\text{revoke}(S,D), \text{Hash}(\text{link}5)) \)

Link 7: \( \sigma_{s'}(U', \text{Hash}(\text{link}6)) \)
rhine 2018-08

Last used Nov 21, 2018
2 months ago

Added Aug 17, 2018
by iphone 8

Revoke this device
Proving External Corroboration

- Alice posts a signature saying she is @theRealAlice on Twitter
- Then posts a hash of that signature to twitter
Link 8:
\(\sigma_s'(\text{twitter: @theRealAlice}, \text{Hash(link7)})\)
How Does Bob Lookup Alice? Idea #1

• He fetches her “sigchain” from the server

• Playback chain from beginning to compute:
  • Signing Keys: \( \{S'\} \)
  • DH Keys: \( \{D'\} \)
  • Per-User-Key: \( U' \)
  • Claimed external identities: \{ twitter: @theRealAlice \}
Idea #1

- What attacks can you think of?
Idea #2

• Download Merkle root from server, and verify explicit signature (i.e., don’t just trust TLS). (Why?)

• Descend the Merkle tree to Alice’s leaf

• Fetch tail of her “sigchain” and confirm the returned sigchain from #1 ends in the advertised tail

• As before
Idea #2: Additional Bookkeeping

- Whenever Bob looks up Alice at time $t_1$ and $t_2$, he asserts the new links fit at the end of the chain.

- Whenever Bob looks up Alice at time $t_1$ and Charlie at time $t_2$, ensures:
  - The global Merkle sequence # has increased.
  - And that the global Merkle root points back to the earlier root via logarithmic “skip pointers”.
Demo

• https://keybase.io/_/api/1.0/merkle/path.json?username=max&last=4000000

• https://keybase.io/max/sigchain
Idea #2: What Other Attacks?

• “Forking attack”
  
  • https://www.blockchain.com/btc/address/1HUCBSJeHnkhrVKVjaVmWg2QtZS1mdfaz

• Sprinkle roots all over the internet

• Odd/Even Attacks
How to Define a Team
Creating a Team

• Alice creates the team “coinco” with two admins, her and Bob.

• Rolls a new team secret: $t$

  • From $t$, generates team public keys:
    • $(s_t, S_t)$ for signing
    • $(d_t, D_t)$ for Diffie-Hellman
    • And a symmetric key for encrypted shared team data

• Encrypts $t$ for $U_A$ and $U_B$
Link 1:

\[ \sigma_A(\text{name=coinco, admins=\{Alice, Bob\}, keys=\{S_t, D_t\}}) \]
Adding a User to a Team

- Alice or Bob can now add Chuck to the team:
  - Admins can make membership changes
  - Non-admins just get to see team secrets
- Adds a sigchain link
- Encrypts $t$ for $U_C$
Link 2:
\( \sigma_B(\text{admins} = \{\text{Chuck}\}, \text{Hash}(\text{link1})) \)
Removing a User

- Admins can remove users, but must re-roll the team keys
Link 3:
\[ \sigma_C(\text{remove}(\text{Alice}), \text{keys} = \{S'_t, D'_t\}, \text{Hash(link2)}) \]
When Else Are Keys Rotated?

• When a team member “resets” their account
• When a team member revokes a device
• When a team member “leaves” a team
Revoking a Device, Revisited

• Whenever team members revoke devices, their per-user-keys re-roll

• Therefore all teams they are in must re-roll their keys

• This can be done **lazily**, just before the next time someone chats, or writes a file for the team
Loading a Team

• Load the most recent Merkle root, and descend to the team’s leaf

• “Play” the team chain forward and ensure:
  • Tail matches what was in the Merkle Tree
  • That all modifications are made by authorized admins
  • All links are signed with keys that were valid for the user at the time of their signature
A New Challenge: Cross-Chain Ordering

- Bobs sees that Alice made a change a team at sequence $m$ in chain $C_{\text{team}}$

- Sees that Alice revoked that device at sequence $n$ in chain $C_{\text{alice}}$

- He needs proof that the first event happens before the second
Loading Teams: Performance

- [https://keybase.io/team/keybasefriends](https://keybase.io/team/keybasefriends)
  - 2400 members
  - 5395 sigchain links
  - ~12MB in transfer size
  - + 8 admins, each with lengthy sigchains
Insight: UI Doesn’t show all 2400 people

- So don’t bother to derive group membership at first
- Just load sigchain links that advertise keys
- Lazy-load membership info
- “Stubbed chain”
Attacks on Teams

• In practice, server coordinates client key rotations
  • Clients audit in background loops that keys are adequately rotated

• Odd/Even Attack
  • Clients probabilistically audit team chain history on the critical path
Key Learnings & Challenges
Key Learning: Username to UID mapping

- UID is just the hash of the username
Key Learning: PUKs

• v1.0 was built without
• Alice’s mobile provisions a new laptop:
  • for all teams Alice is in:
    • Reencrypt team secret for laptop
• Rekey races Alice backgrounding the app
• Can resulting viral data loss across devices!
Key Challenges

- Immutable append-only storage
- Shipping client code on 5 platforms
- Clients must distrust the server, and sometimes just intentionally break
- User Education / Account Resets
Hi Max,

My server crashed and burned, and I lost my paper keys and Jalepeno. Is there any way I can provide some sort of ID verification to get my keybase account back? Thanks!

Sincerely,

J
In Sum...

- Key problem: multi-device with instant access on new device
  - Solution: Per-user-keys
- Users are chains of device additions/removals
  - All devices are equally powerful
- Teams are chains of user additions/removals
  - All admins are equally powerful
- From there, build a shared secret key for teams that rotates on revocation or member removal.