Better authentication: password revolution by evolution

Daniel R. Thomas and Alastair R. Beresford

22nd Security Protocols Workshop

UNIVERSITY OF CAMBRIDGE

Daniel: 5017 A1EC 0B29 08E3 CF64 7CCD 5514 35D5 D749 33D9
Alastair: 9217 482D D647 8641 44BA 10D8 83F4 9FBF 1144 D9B3

This is a crazy idea I had after last year's SPW.

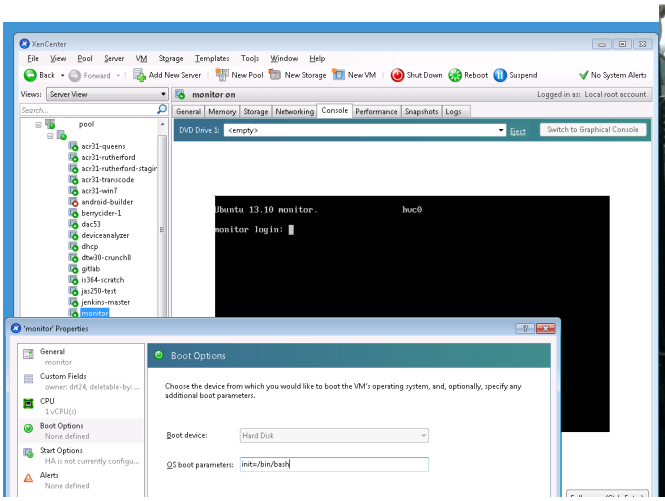
Idea: Use public key based tokens for password login

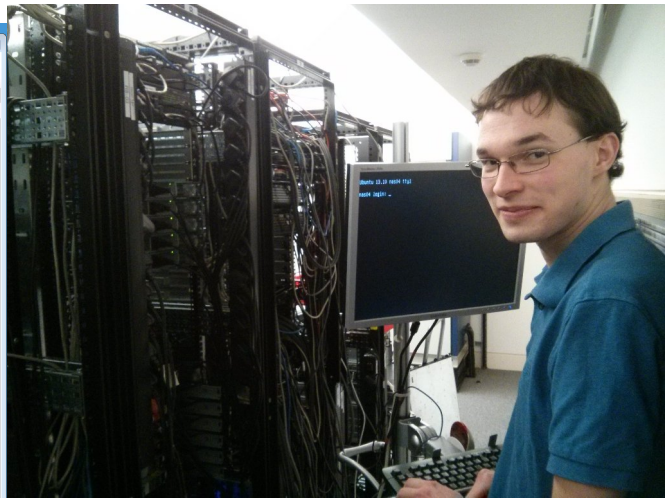
- One time token based authentication using public key cryptography
- Allow machines to provision themselves from public data and allow login
- Inspiration from Monkeysphere and Google Authenticator

We want to replace passwords in a scalable way, and so want to use public key cryptography rather than lots of shared secrets
A one time token using public keys rather than symmetric keys could be used in existing password prompts but verified with a different verification function
If we use public information for login verification then machines can work out whether someone should be allowed to login without any prior contact with an administrator
Monkeysphere does a similar public key infrastructure system for ssh and Google Authenticator uses one time tokens for authentication

Motivation: Passwords don't scale

- when you have many machines and don't want compromising one to compromise the others.
- per machine, per user passwords do not scale ($\mathrm{O}(mn)$)
- Solutions for ssh but what about when networking broken?





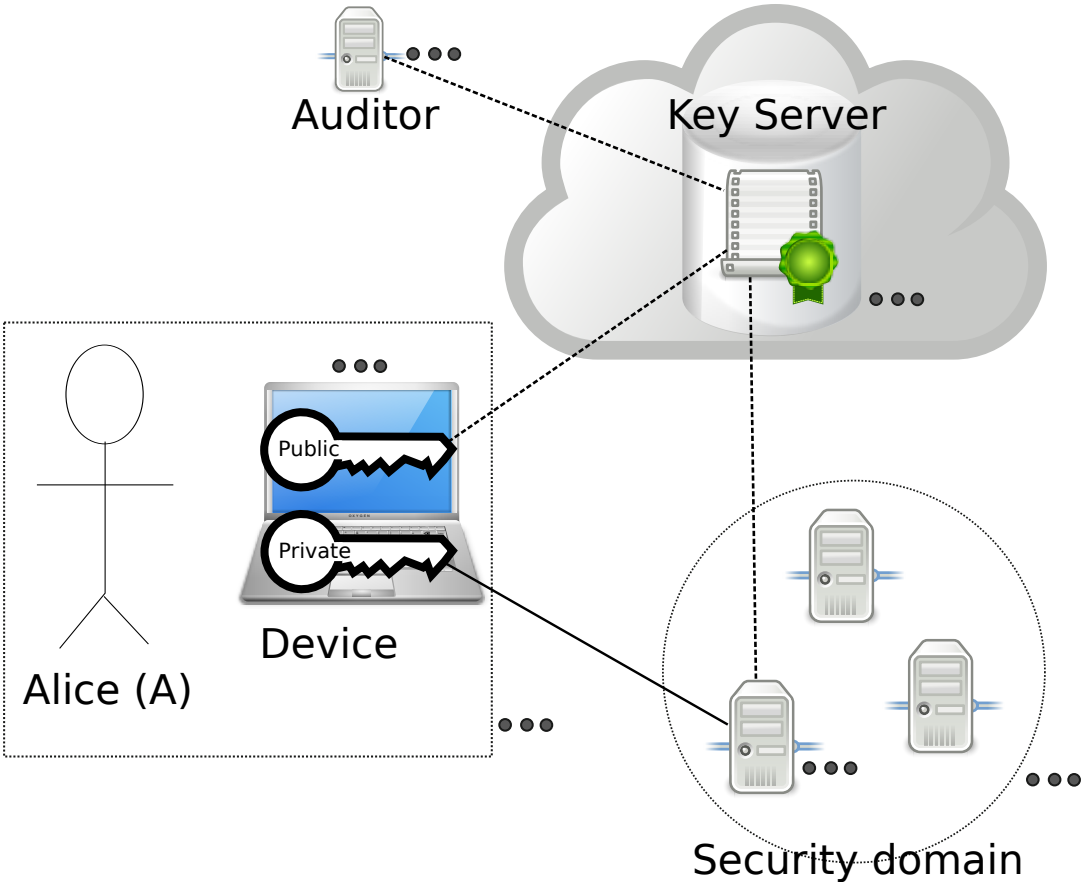
We have a bunch of servers and VMs with entirely public configurations as that makes it easier for them to self-configure. We can use monkeysphere to do authentication over ssh in a scalable way. But what about passwords? Hashes could be brute forced if one machine was compromised and harder to provision servers if they need secrets. Per machine, per user passwords just doesn't work.
The picture on the left shows an attempt to login to a VM on the console in XenCenter, since there is no password it must be rebooted with the kernel options changed to `init=/bin/bash` to put it in single user mode.
The picture on the right shows an attempt to login to the console on a physical machine, again it needs to be rebooted into single user mode but that involves pressing escape at the right point in the boot cycle.

Replace passwords

- Something we could type in
- Something we don't have to remember
- Something which does not require sharing secrets
- We can use things we already have

We want something which scales to many machines and users but which is compatible with existing password prompts. We have devices which can do some cryptography and store keys and we have networking.

Public keys can be distributed securely



I am going to wave my hands a bit and say there is ongoing research on solving PKI problem such as Certificate Transparency and so while the details of exactly how it works are important I am mostly going to refer you to the paper and to related work.
For all these things there are many of them, many users with many devices, many security domains with many end points.
A user has some devices, each of which has a key pair. They send their public key and signatures on it to some key servers which maintains a database of these keys and signatures and an append only signed log. The behaviour of the key server can be verified by auditors which examine the log and the database.
End points can subscribe to relevant signatures and keys from the key servers.
Users register public keys with each security domain to which they can then authenticate using their private key.
The Auditor is a server run by someone like the EFF which verifies the behaviour of the key server and if the key server lies to anyone then the auditor can detect this.
For our purposes with only a few hundred thousand technical users it scales easily, since we don't need a web of trust it scales much better than some other PKI systems. In term of storage it easily scales to facebook and authentication becomes a more distributed process which hopefully improves scalability.

A can authenticate by sending a token

SOTTA: Simple one time token authentication

Alice (A) can authenticate to the end point (S) in one step by sending a token:

$$A \rightarrow S : A, \{D||\lfloor t \rfloor\}_{K_A^{-1}}$$

D is the domain identifier (e.g. `dtg.cl.cam.ac.uk`) and $\lfloor t \rfloor$ is the quantised time (e.g. to the nearest minute)

Here Alice (A) authenticates to the end point (S) by saying that she is Alice and giving a token which proves it, D is the identifier for the security domain which S is in and we also include the quantised time.

Signature length is too long

Username: drt24
Password: Xq8xdTBjJHunpIC64pBm6Q94wdlZkwiyrilTJgx5b4oFEYHUiD
ZZXIL4ouSxNW6YD3y8IsZSNDKNYKA7sYWfUi
(1 minute 20s to input password - and I got it wrong)

Here we have an example of an authentication using this protocol, the ‘password’ here is the right size to be a DSA signature of the domain and quantised time. Clearly this is problematic because it takes me 1 minute 20 to type that out and I typed it wrong.

Signature length is too long

Need resistance to offline brute force $\Rightarrow$ 128 bits of security.

Bits	Bytes	numeric [0-9]	alphabetical [a-z]	Alphanumeric [A-Za-z0-9]	Algorithm
32	4	10	7	6	
64	8	20	14	11	
80	10	25	18	14	
128	16	39	28	22	
160	20	49	35	27	
256	32	78	55	43	Minimum
320	40	97	69	54	BSL?
512	64	155	109	86	DSA
1024	128	309	218	172	
3072	384	925	654	516	RSA

Table : Encoding sizes for different bit lengths

The problem is that the signature length is too long. This table shows the number of characters required to encode signatures of different lengths. RSA signatures would be 516 characters. DSA signatures are 86 characters – and there are severe problems with DSA. Ideally we want a public key signature scheme with a signature size of at most 256 bits which is then ‘only’ 43 characters. The BSL scheme might provide this but it does not seem that popular.

The long random string can be automatically input

AOTTA: Automatic one time token authentication

Some of the time the long random string can be automatically input

- bluetooth keyboard
- Copy & Paste
- QR code
- audio networking

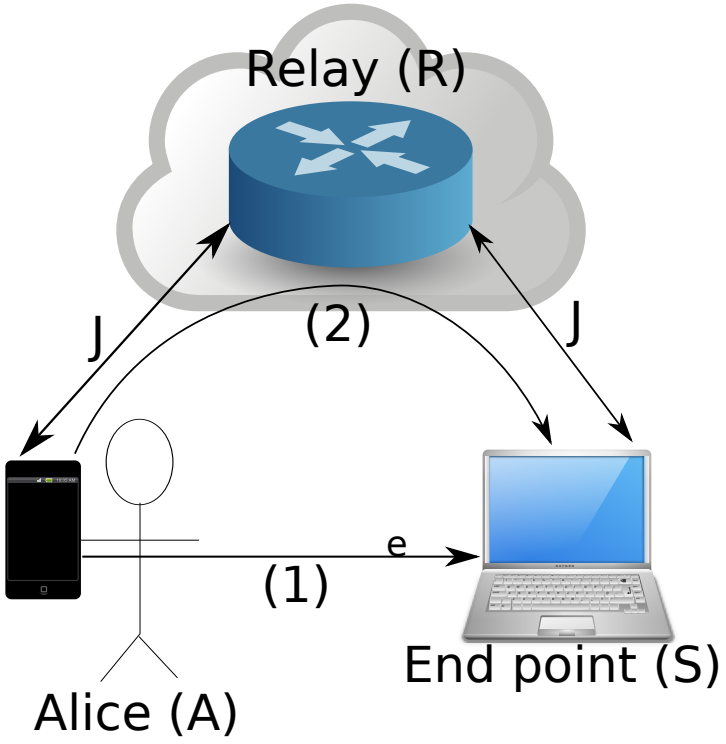
There are a variety of tactics we can use to automate the process of inputting the token and so avoid typing it in. This can be particularly simple if the device with the keys is being used to connect to the end point which requires authentication as then it can be copied and pasted.

Online connectivity enables short tokens

OOTTA: Online one time token authentication

$$A \vec{e} S : s \qquad (1)$$

$$A \rightarrow S : \mathrm{E}_{\mathcal{K}}(\{D||\lfloor t \rfloor\}_{K_A^{-1}}) \qquad (2)$$



If the device with the keys and the end point both have a working network connection then we can use PAKE to connect the two over a secure connection authenticated with a short token sent locally and then send the public key token as before.

We can deploy this

- Can bootstrap with existing PGP / GPG infrastructure used for monkeysphere
- Authenticate to what?
- Key servers
- Relays
- Devices

Authenticate to servers and websites which have password prompts at present.
Google, ICANN etc. might want to run key servers
Relays could be run by companies such as Google or self hosted
Phones, laptops, desktops, smartwatches etc.

Better authentication: password revolution by evolution

We can use public key cryptography to produce one time tokens allowing authentication through typing on a keyboard.

- Daniel R. Thomas drt24@cam.ac.uk
  5017 A1EC 0B29 08E3 CF64 7CCD 5514 35D5 D749 33D9
- Alastair R. Beresford arb33@cam.ac.uk
  9217 482D D647 8641 44BA 10D8 83F4 9FBF 1144 D9B3

Acknowledgements

- Useful feedback received from Andrew Rice and Robert Watson. Suggestion of BSL scheme from Markus Kuhn.
- Bluetooth logo from open icon library
- Phone picture from openclipart
- Computer laptop from open icon library
- Copy from open icon library
- Google Authenticator logo from Google.
- Router image from open clip art
- Monkeysphere logo based on wikimedia commons image
- Networked server picture from open icon library
- Paste icon from open icon library
- Green seal from open clip art
- Key diagram based on one from open clip art