authreq

Using mobile devices' hardware-backed keystore for universal authentication

Akos Szente (2094613s)
akos@szente.info

https://authreq.szente.info/

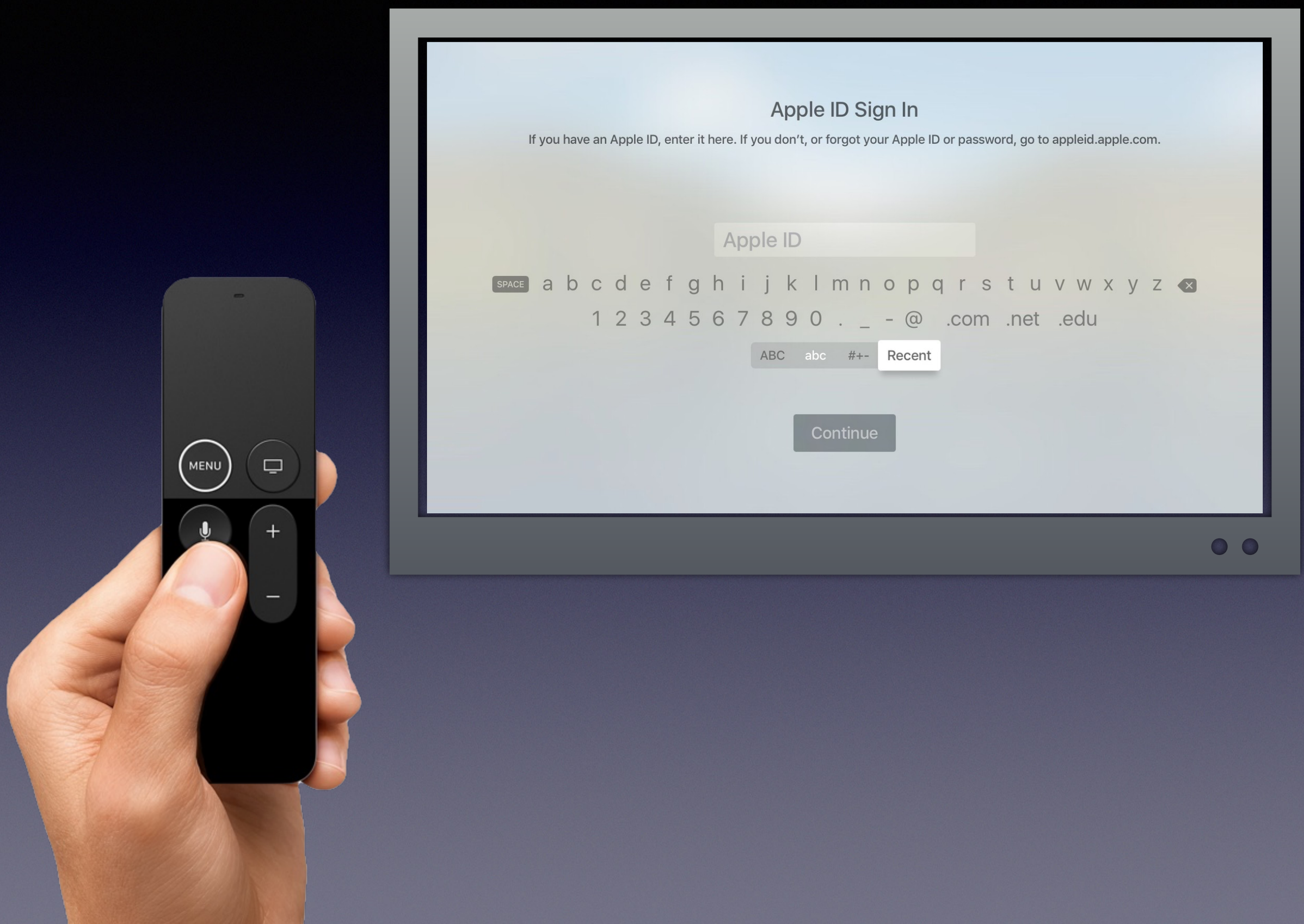
A lightweight **authentication scheme** that

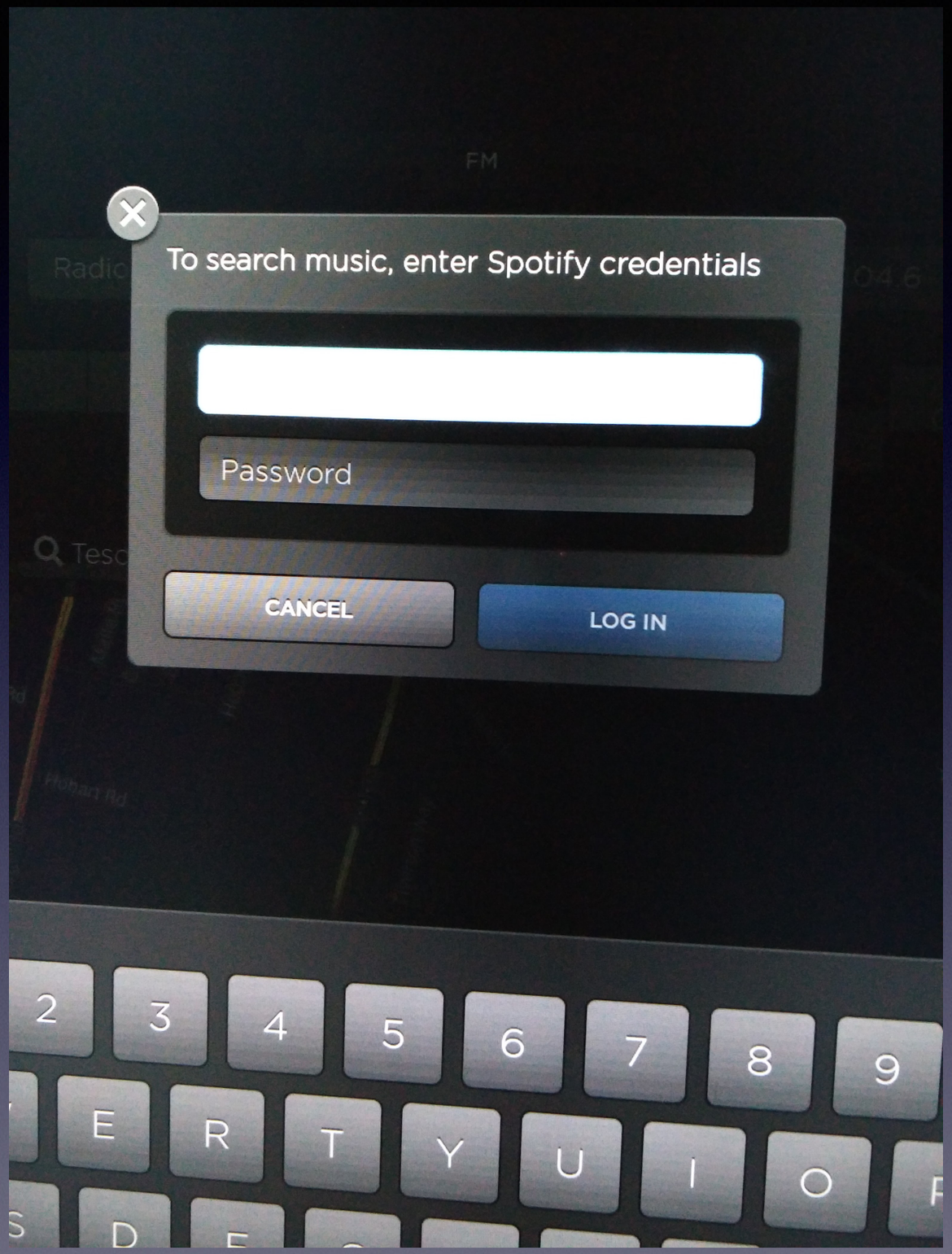
uses **native e-signature capabilities** of iOS

to confirm legitimate **user identity**

for third-party services.


NETFLIX
Email
Password
Sign In
Restore Purchase
MENU


Apple ID Sign In
If you have an Apple ID, enter it here. If you don't, or forgot your Apple ID or password, go to appleid.apple.com.
Apple ID
SPACE a b c d e f g h i j k l m n o p q r s t u v w x y z
1 2 3 4 5 6 7 8 9 0 . _ - @ .com .net .edu
ABC abc #+- Recent
Continue
MENU



https://speakev.com/threads/spotify-asking-for-login-details.14213/

Passwords made sense...

- ...when interaction was primarily via **keyboard**
- ...when there was **nothing better available**

(early 1960s)

Other issues

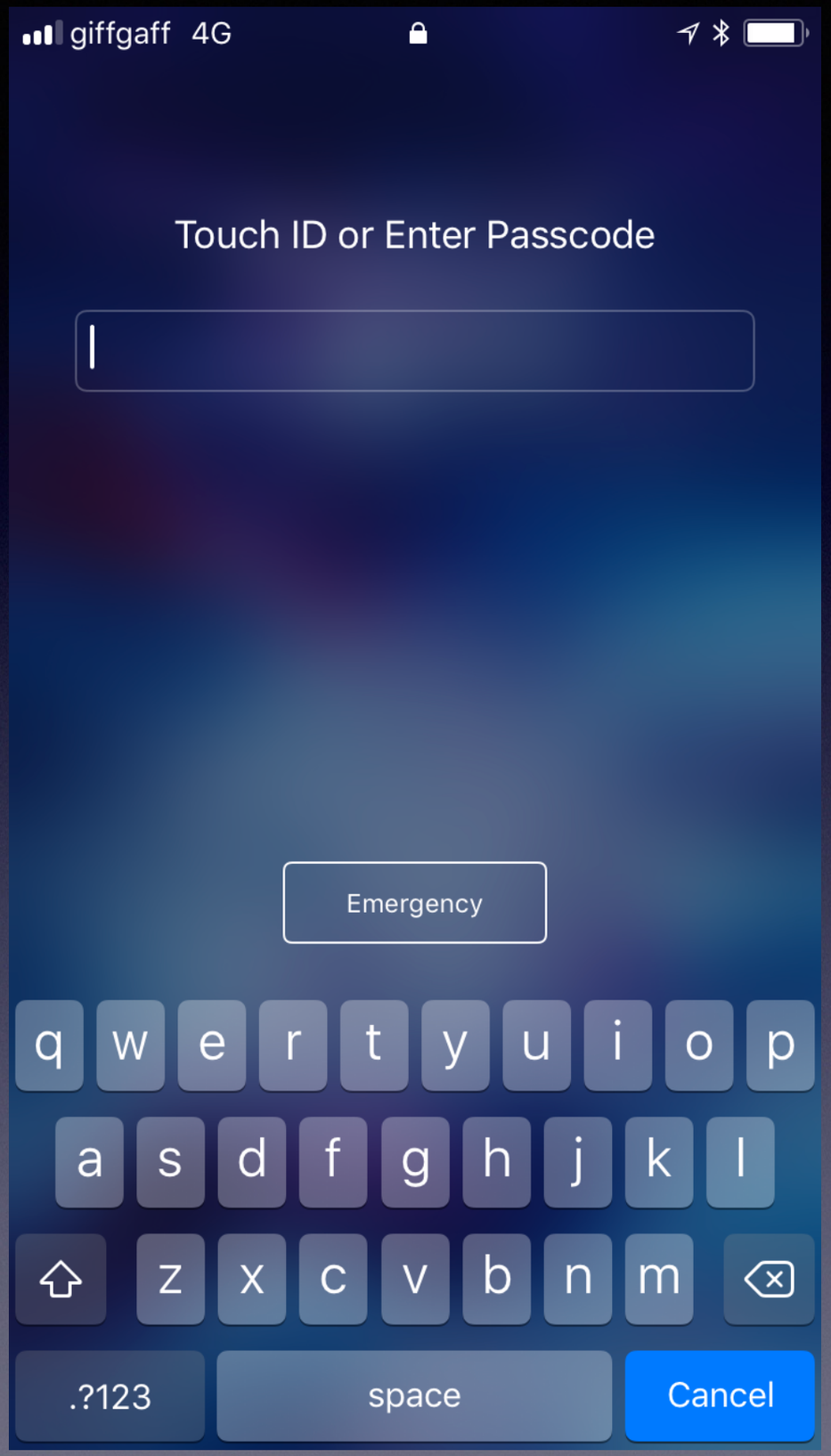
- **Not ephemeral** (capture once, replay forever)
- **Password re-use** (40% directly, 71% indirectly)
  [Das et al., 2014]
- **No revocation** (must change manually everywhere)
- **Social engineering**

If not passwords, then what?

Knowledge (Password)	**Possession** (Token)
Inherence (Fingerprint)	***Combination of these***

Passcode

Fingerprint

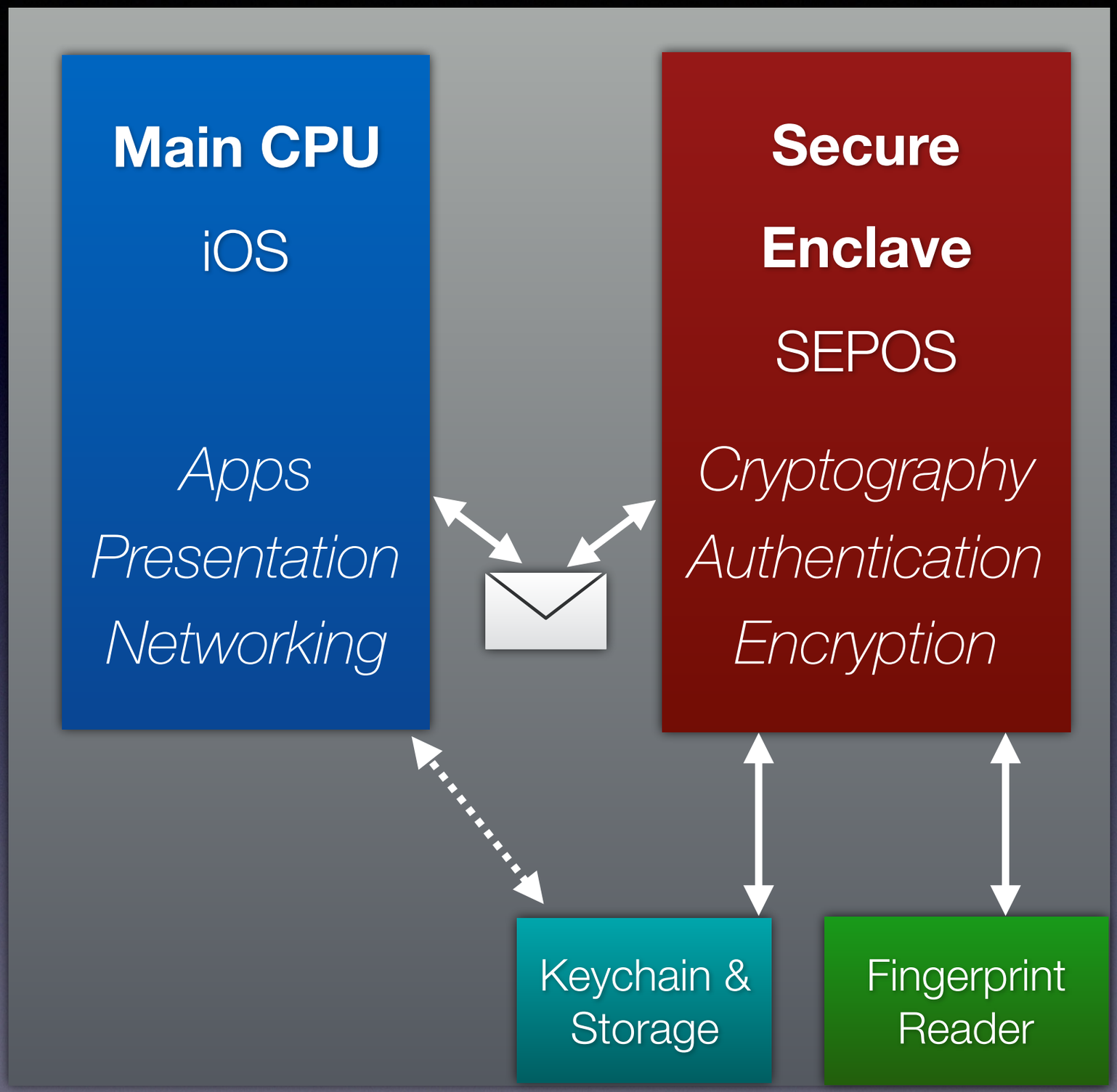


****

Password

+

Fingerprint



- User requested unlock.

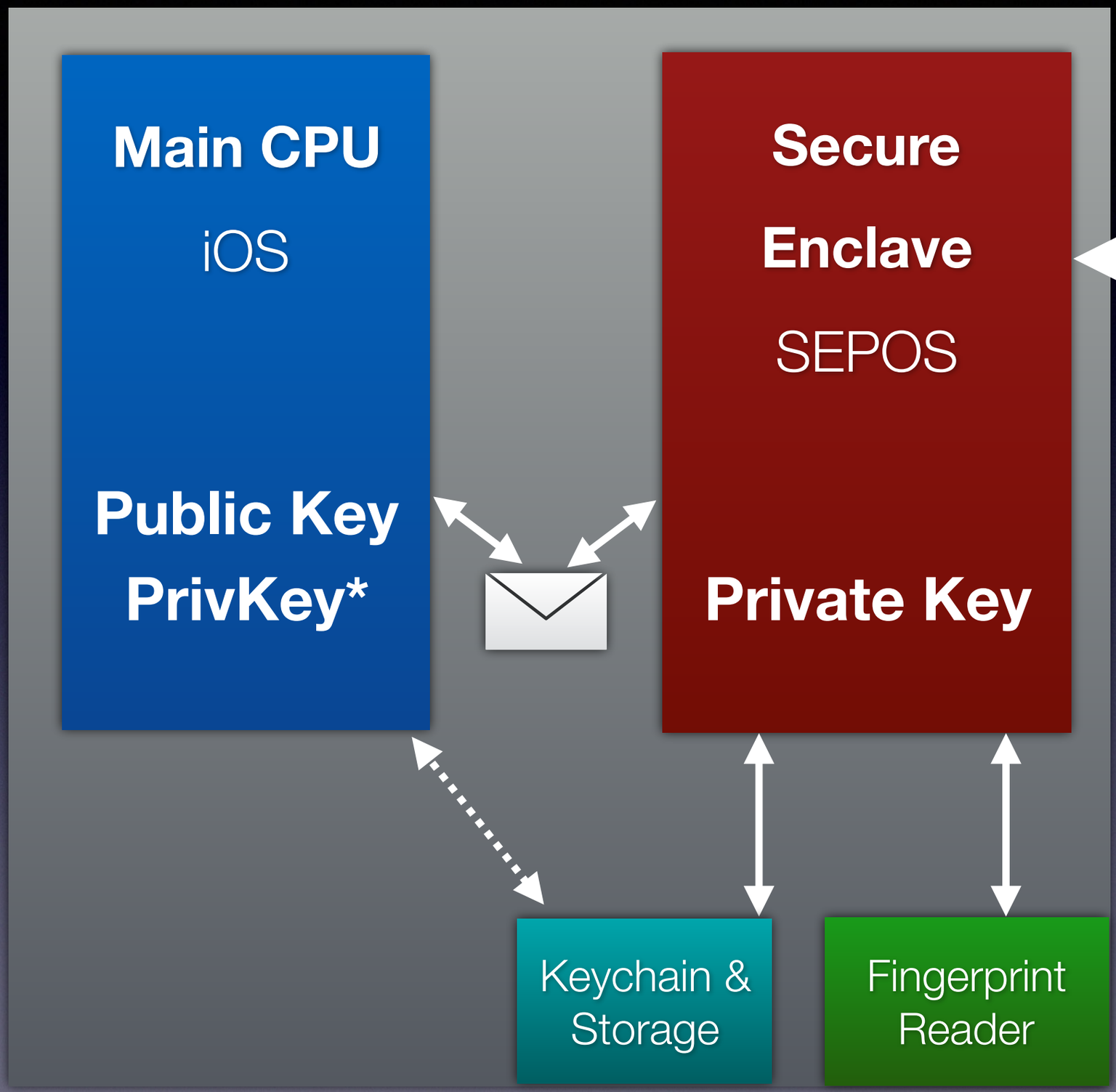
- Tell them to use either Touch ID or passcode.

- I've received an invalid fingerprint. Tell them to try again. (3x)

- I've disposed of the saved passcode. Tell them that they must enter passcode.

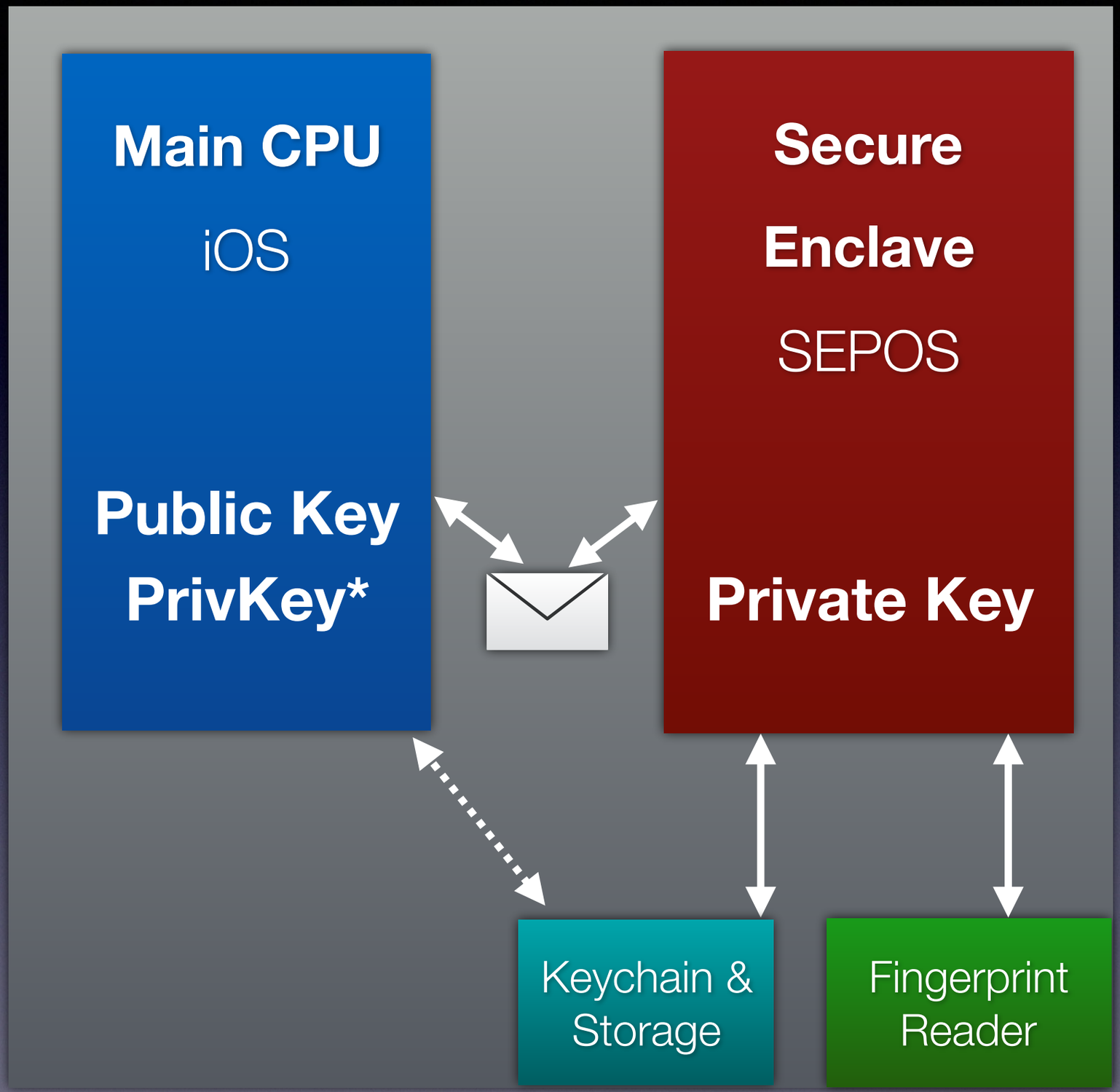
- User entered 328000.

- That's valid. I've unlocked Keychain and Storage, and saved the passcode. They can now use Touch ID. Proceed to Home screen.



Keystore

Within



- Please sign the following digest with PrivKey*.

- This key is protected. Tell the user to use Touch ID for local authentication.

(User authenticates via Touch ID)

- The signature for the digest is [234aafd0...]

Overview

- **Generate** a pair of public and private keys in Secure Enclave and set up local authentication enforcement

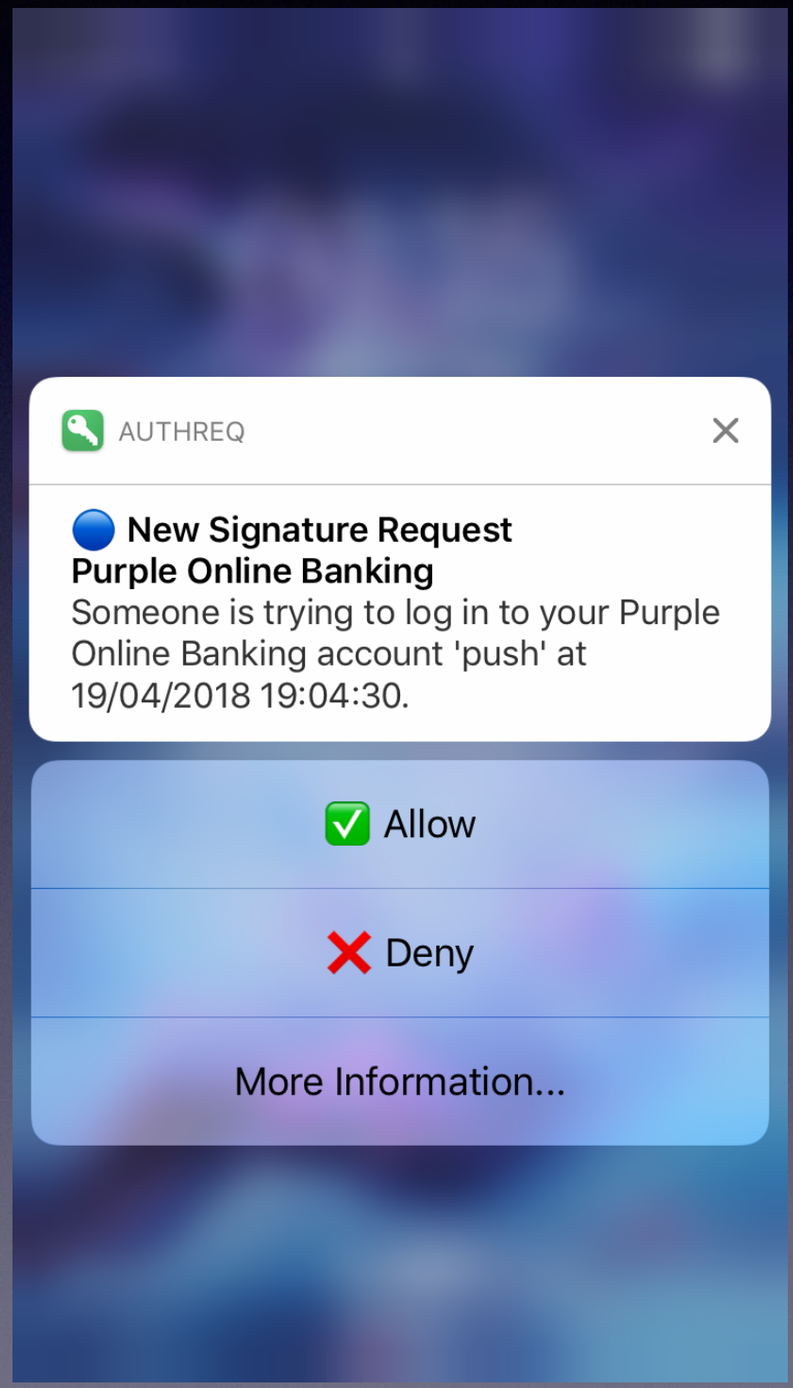
- **Share** the public key with third-party services

- Whenever authentication is necessary, these services can send an **Authentication Challenge**

- **Signing** the challenge requires device possession and local authentication, thus it efficiently confirms legitimate user identity.

Authentication Challenge

- **Message_id** *(523452)*
- **Textual content fields**
  - Subtitle *(Purple Online Banking)*
  - Short_title *(Login Attempt)*
  - Body *(Someone is trying to log in...)*
- **Expiry** *(1519387343)*
- **Nonce** *(b5bc3bb46e940ce73591b2...)*
- **Response_url** *(https://s02.szente.info/authreq-srv/callback.php)*



Reply

- **Message_id** *(523452)*
- **Signature** *(304602210 09c53817d9222017713f340c5...)*
- **Public_key** *(-----BEGIN PUBLIC KEY----- MFkwEwYH ...)*
- **Token** *(1daa2201782134eab20...)*

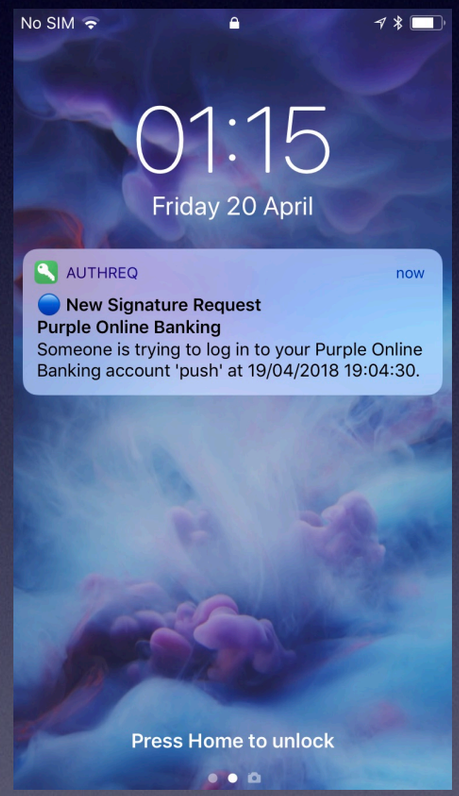
for enrolment

Communication

Apple Push
Notification Service

Authentication challenge

Push Notification



iOS Client

Enrolment,
Authentication challenge

URL Scheme
(QR code, clickable button, etc.)

Server

Reply

HTTPS

Add to authreq

DEMO

What You See Is What You Sign

- **Landrock** and **Pedersen**
- Binary data can have multiple interpretations, but it is the binary data that the user signs
- Must leave no way to alter semantic interpretation of original message, so that humans can be certain that what they see is what they sign

What You See Is What You Sign

- Fields distilled into a canonical format
- Bencode

  d4:body138:Someone is trying to log in to your Purple Online Banking account 'push' from Glasgow, United Kingdom at 23/02/2018 07:02:23. Is this you? 8:category17:challengecategory6:expiryi1519387343e10: message_idi523452e5:nonce64:b5bc3bb46e940ce73591b2f180cc28cb29e22cbd211 895a22e0aef615bf71c1212:response_url48:https://s02.szente.info/authreq-srv/ callback.php11:short_title13:Login Attempt8:subtitle21:Purple Online Banking5:title26:New requeste
- Message Digest - Hash

  71f182c099317c 4f86ecae7edc315881bb8f47a45f682d8a1f7d6cc51531573f4 295d984756ae7e92dbb7d220bbdc932.

Goals

- Authentication that immediately makes sense
- Mitigate the presented issues of passwords
- More secure than popular 2FA solutions
- Quicker and more usable

Implementation

- iOS Client - Swift 4, iOS 11
- Server - PHP, OpenSSL
- Service SDK - PHP
- Sample Service - PHP, Yii

Cryptography

- 256-bit ECDSA (secp256r1)
- Digests created via SHA-384
- Client: Apple Security Framework
- Server: OpenSSL

Usability

- Mimics native applications of iOS
  - Rich Notifications, 3D Touch, Taptic Engine, URL Scheme
- Simple mental model: ***approval***
- Using **digital signature** and **PKI** technology without mentioning either

Evaluation

- **Comparing to the Five Problematic Properties of Security** [*Why Johnny Can't Encrypt* (Whitten et al.)]

- **Quality Coefficient** [*Quantifying the Quality of Web Authentication Mechanisms — A Usability Perspective* (Renaud)]

- **User Study** [*Two-factor or not two-factor? A comparative usability study of two-factor authentication* (Cristofaro et al.)]

Five Problematic Properties of Security

[*Why Johnny Can't Encrypt* (Whitten et al.)]

- Unmotivated User Property
- Abstraction Property
- Lack of Feedback Property
- Barn Door Property
- Weakest Link Property

Quality Coefficient

[*Quantifying the Quality of Web Authentication Mechanisms — A Usability Perspective* (Renaud)]

Dim *& Aspect*	*Coeff.*	*Reasons*
Accessibility	*ad=0.41*	
Special Requir.	0.33	Application
Convenience	0.25	Fairly quick, but still adds another step
Inclusivity	0	Accessibility features of iOS work with app
Memorability	*md=0*	
Retrieval	0	Nothing to recall
Meaningfulness	0	N/A
Depth of proc.	0	N/A
Security	*sd=0*	
Predictability	0	Signature prediction requires private key
Abundance	0	High entropy
Disclosure	0	The private key is adequately protected from disclosure
Vulnerability	*vd=0*	
Confidentiality	0	Signature differs every time. Cannot be reused
Privacy	0	Leaks have no consequences to privacy
Breakability	0	Not prone to social engineering and research based attacks; ECDSA widely considered safe

Passwords	8.71
SMS OTP	10.71
Card Reader	9.575
Google Authenticator	10.76
Authreq	12.385

User Study

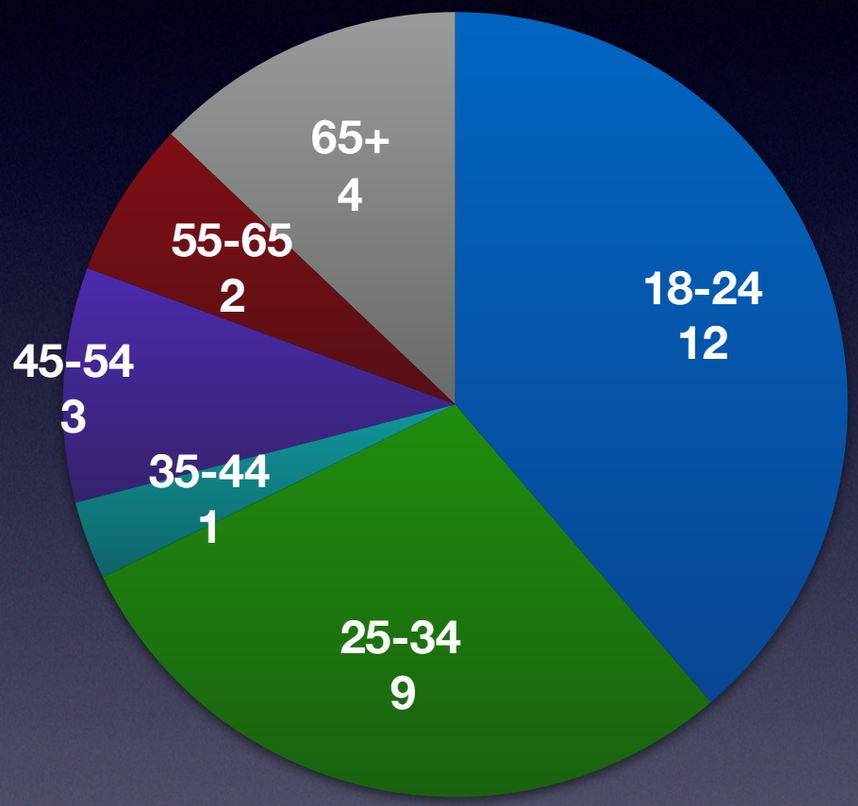
[Two-factor or not two-factor? A comparative usability study of two-factor authentication (Cristofaro et al.)]

- **Cristofaro et al.:**
  - **Stage 1:** One-on-one interviews about experiences with 2FA
  - **Stage 2:** Quantitative MTurk study, questionnaire about past experiences with 2FA
- 3 factors: ease of use, trust, cognitive effort
- Four mechanisms were all perceived as highly usable
- Perceived trustworthiness not negatively correlated with ease of use and required cognitive efforts

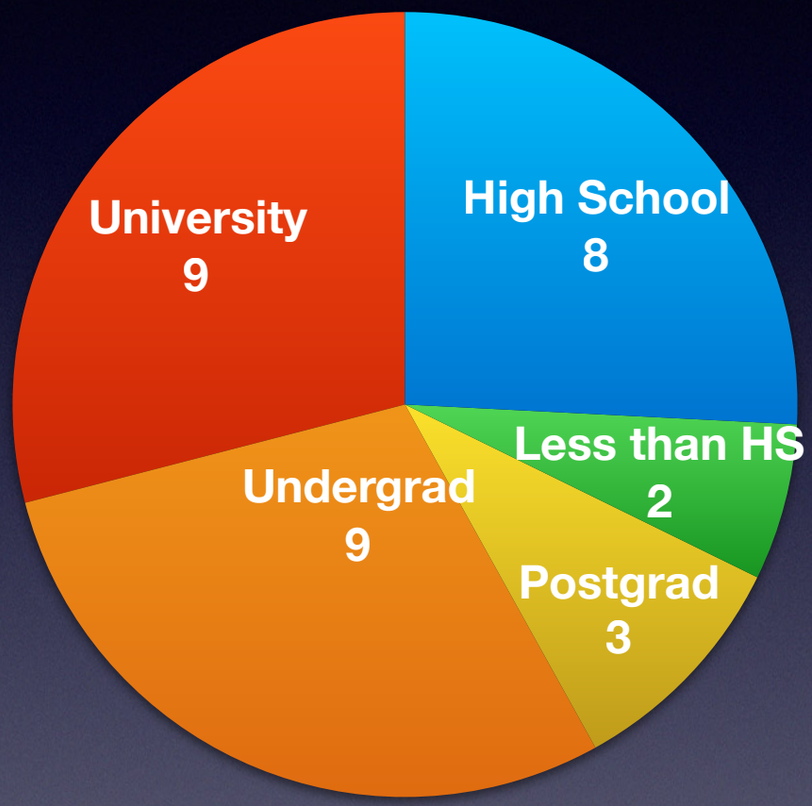
User Study

- **Similar questions and methods as Cristofaro et al.**
- **Measured and compared 4 2FA mechanisms**
  - SMS OTP, Card Reader, Google Authenticator, Authreq
- **Stage 1:** Questionnaire and interview about background and past 2FA & computer experiences
- **Stage 2:** User study with direct observation
- **Stage 3:** Questionnaire about each 2FA mechanism

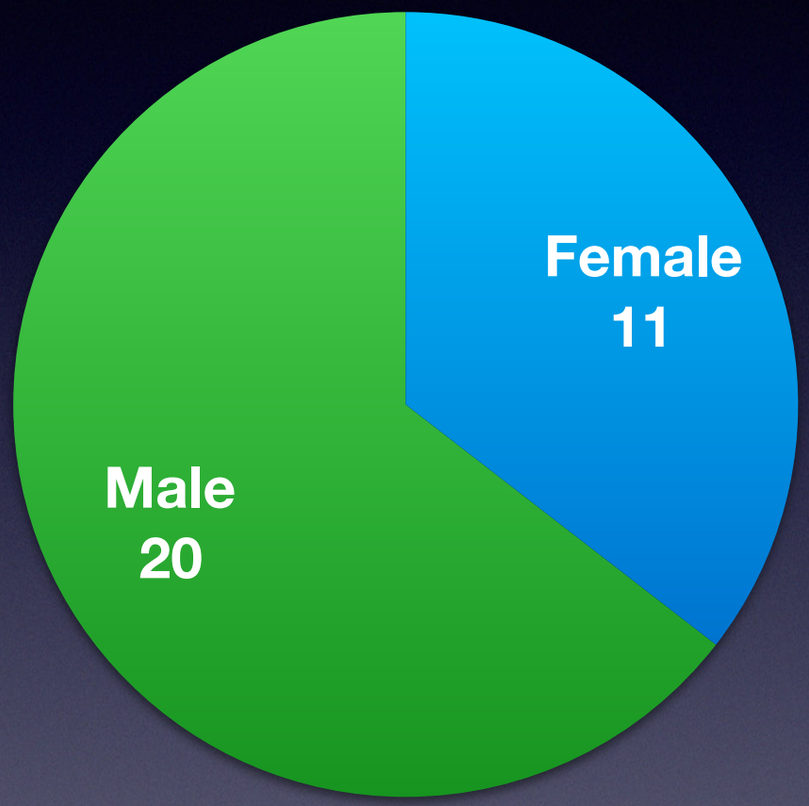
Demographics



Age



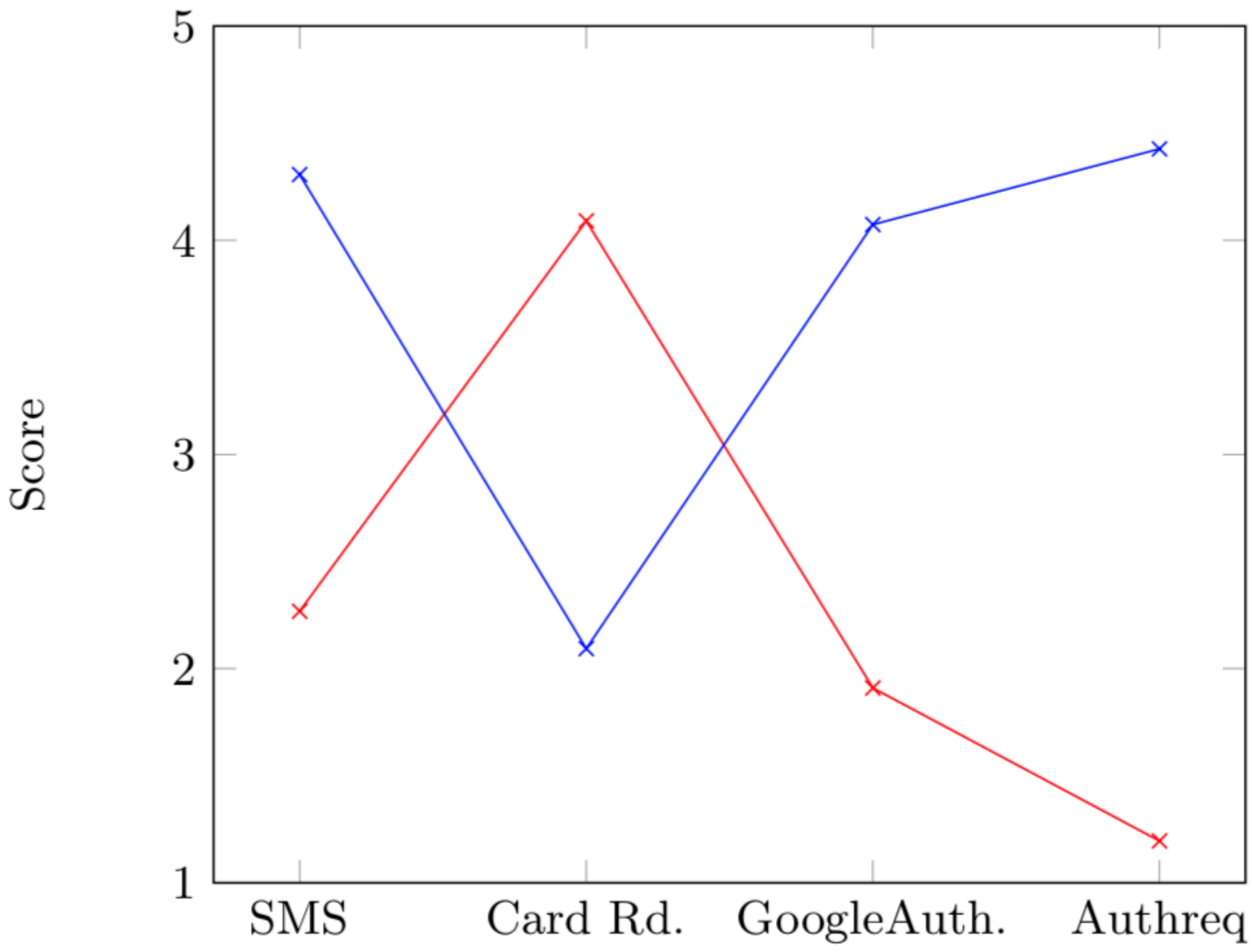
Studies



Gender

31 participants

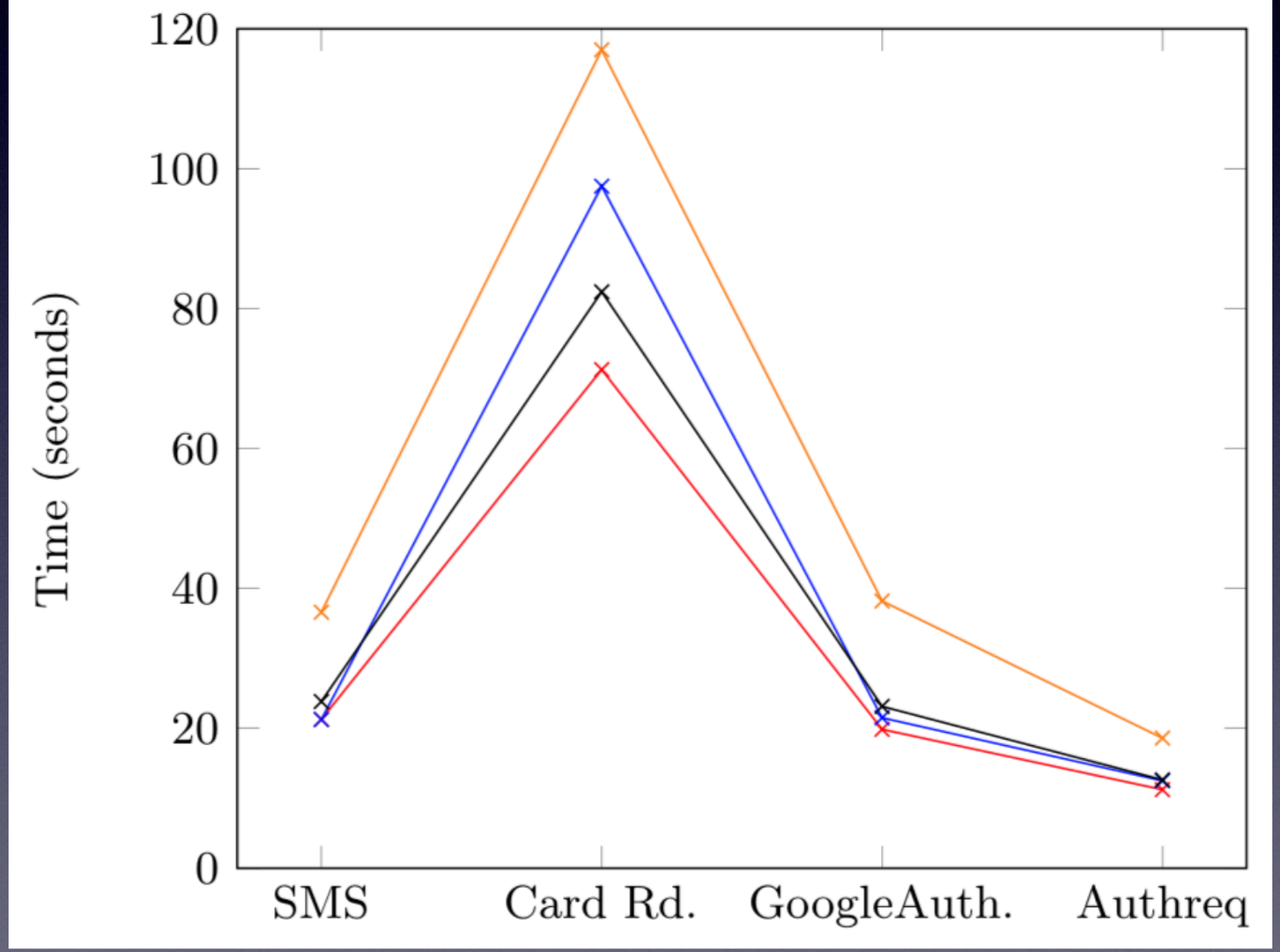
Technology	*Ease of use and trust*	*Cogn. effort*
Card Reader	4.09	2.09
SMS	2.26	4.30
Google Auth.	1.90	4.07
Authreq	1.19	4.42



"People do not generally sit down at their computers wanting to manage their security; rather, they want to send e-mail, browse web pages, or download software, and they want security in place to protect them while they do those things."

–Whitten et al.

Technology	*18-34*	*35-54*	*55+*	*Overall*
SMS	21.28	21.25	36.6	23.83
Card Reader	71.28	97.50	117.00	82.40
Google Authentic.	19.85	21.5	38.20	23.13
Authreq	11.23	12.5	18.60	12.63



Other results & discussion

- Card reader is perceived as most secure
    - Negative correlation with ease of use and cognitive efforts after all?
        - “reassuring”, “100% secure”
        - Accessibility issues
- SMS: de-facto standard, basis of comparison
- Google Authenticator: timer is confusing, but offline

Other results & discussion

- Authreq: “intuituive”, “really fast”
- Most perceived fingerprint authentication as secure, some called it weird
- Not having to open iOS to use
- Similarities with Apple Pay reassuring
- Only works on iPhone

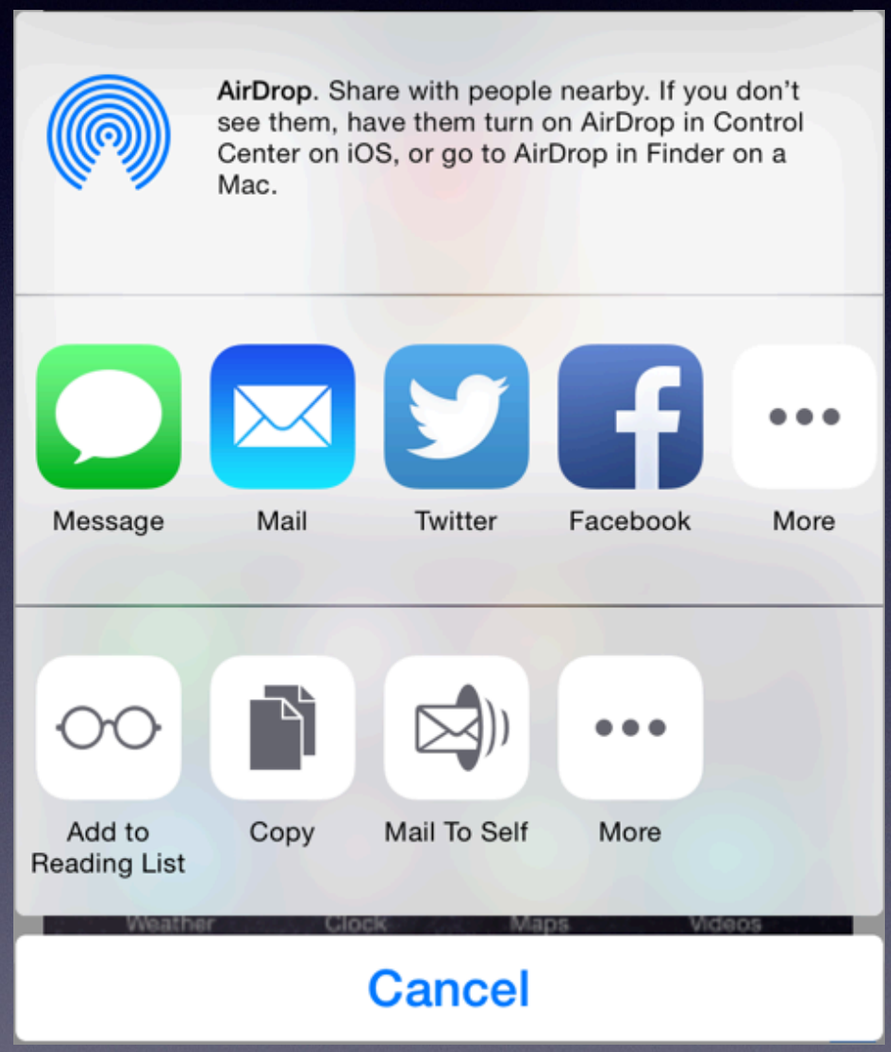
Why not 1FA?

- Registration without password not solved
- Key replacement not solved (e-mail?)

- Current model would not prevent mass login attempts (add another layer?)
- **Future work**

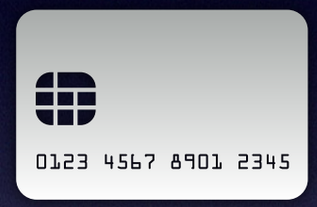
Future Work

- Adding new challenge delivery and reply channels
  - NFC
  - SMS
  - Share Sheet, clipboard as failsafe



Future Work

- Adding new scenarios
  - 3-D Secure
  - EMV (Chip & Pin) transactions
  - Withdraw cash without card
  - Pull printing
  - Opening doors

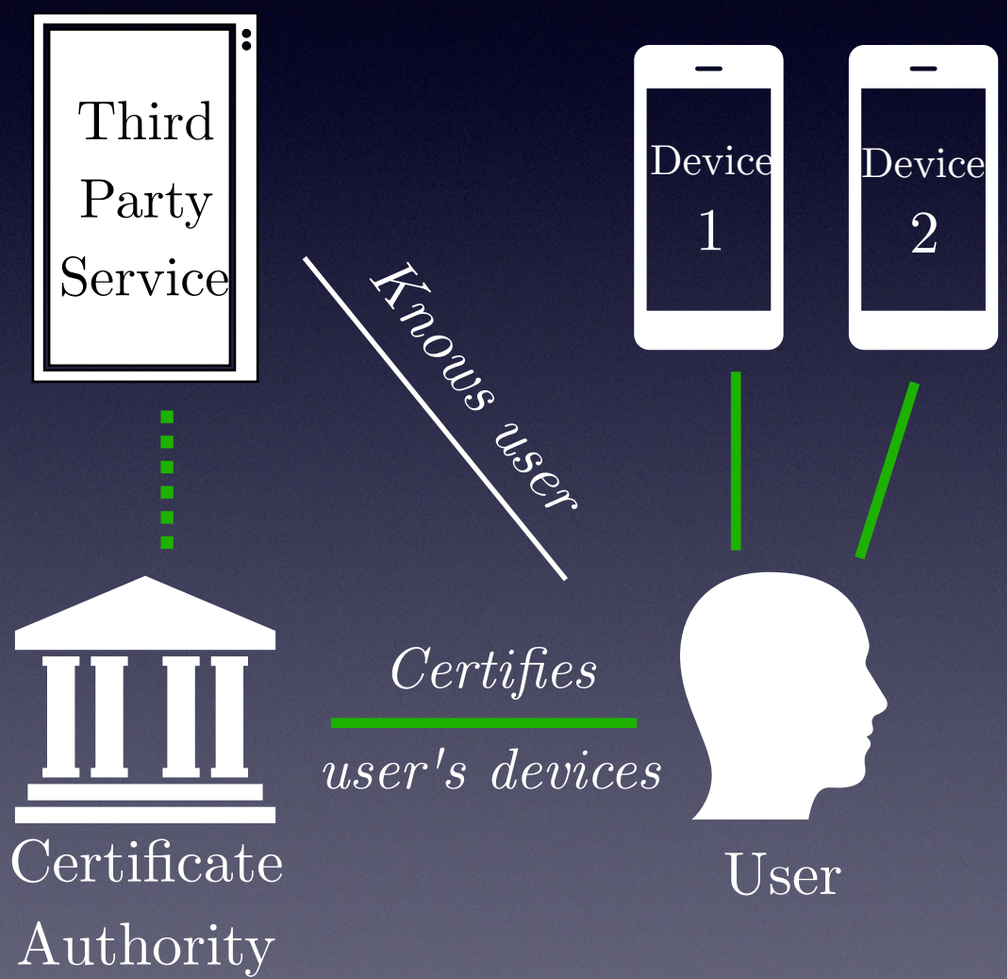


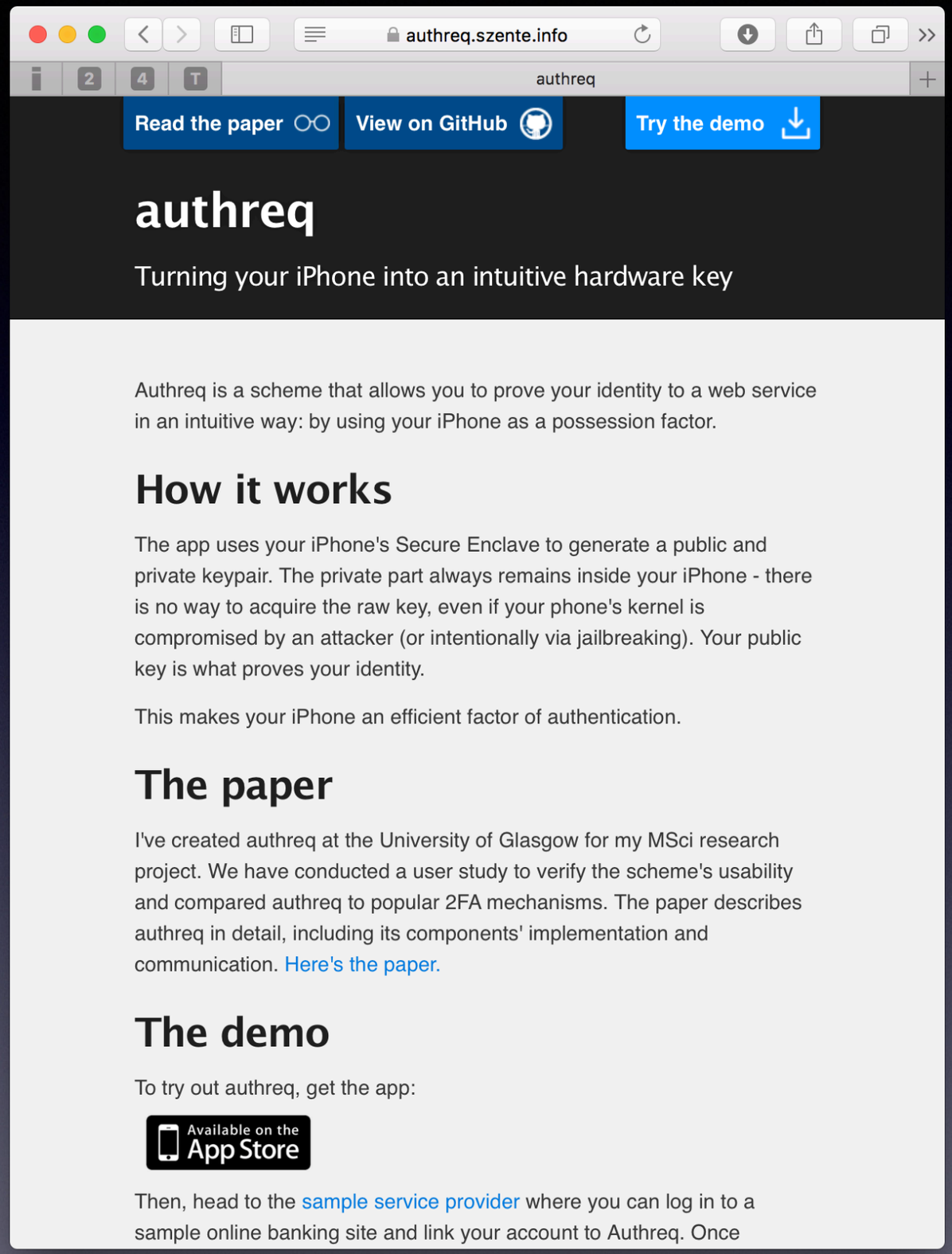
Future Work

- **Integration with Public Key Infrastructure**
  - Currently no CA
  - No way to identify signer without having acquired their public key first
  - Timeframe at enrolment where attacker could MDM and enrol on someone's behalf
  - No key revocation

Future Work

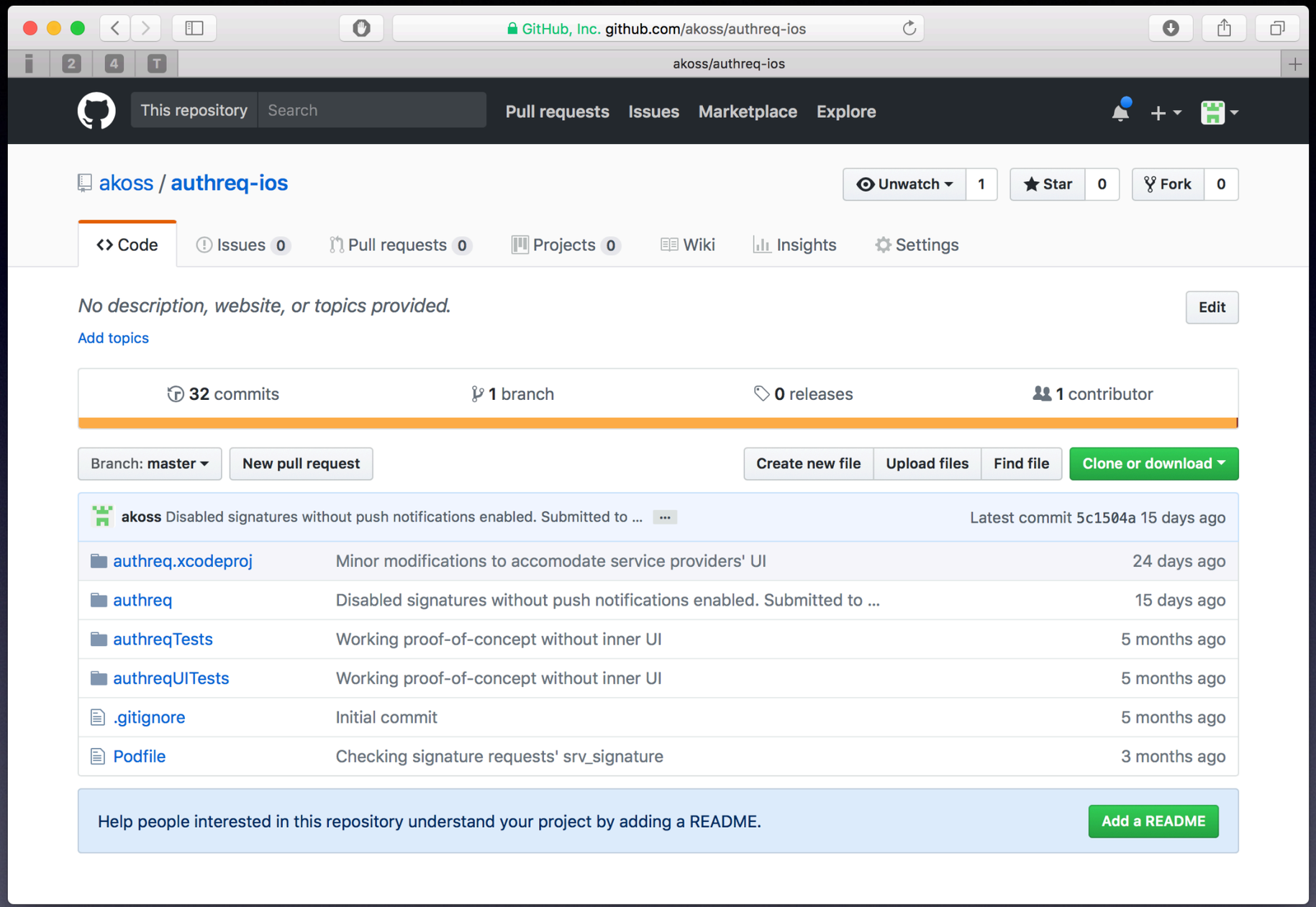
- **Integration with Public Key Infrastructure**
  - Secure Enclave should generate a CSR
  - Certificate Authority would create certificate and offer centralised verification and revocation
  - Key replacement - linking one user's devices





https://
authreq.szente.info

- Dissertation
- Source Code
- Instructions
- iOS App
- Fully working demo environment
- (These slides)


Fork on GitHub
GitHub, Inc. github.com/akoss/authreq-ios
akoss/authreq-ios
This repository
Search
Pull requests
Issues
Marketplace
Explore
akoss / authreq-ios
Unwatch 1
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Insights
Settings
No description, website, or topics provided.
Edit
Add topics
32 commits
1 branch
0 releases
1 contributor
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
akoss Disabled signatures without push notifications enabled. Submitted to ...
Latest commit 5c1504a 15 days ago
authreq.xcodeproj
Minor modifications to accomodate service providers' UI
24 days ago
authreq
Disabled signatures without push notifications enabled. Submitted to ...
15 days ago
authreqTests
Working proof-of-concept without inner UI
5 months ago
authreqUITests
Working proof-of-concept without inner UI
5 months ago
.gitignore
Initial commit
5 months ago
Podfile
Checking signature requests' srv_signature
3 months ago
Help people interested in this repository understand your project by adding a README.
Add a README

MIT License

Recap for Q&A

- Password Troubles
- iOS & Secure Enclave
- Authreq Scheme
- DEMO
- WYSIWYS
- Implementation
- Evaluation
  - Five Problematic Properties of Security
  - Quality Coefficient
  - User Study
- Why Not 1FA
- Future Work

authreq

Using mobile devices' hardware-backed keystore for universal authentication

Thank You

Akos Szente (2094613s)
akos@szente.info

https://authreq.szente.info/