



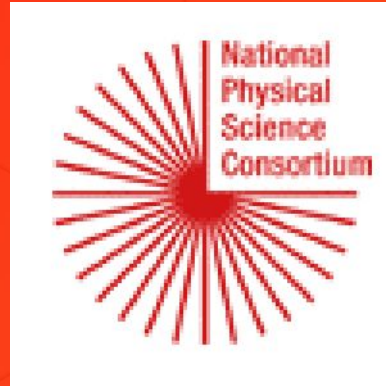
Multi-Factor Key Derivation Function (MFKDF)



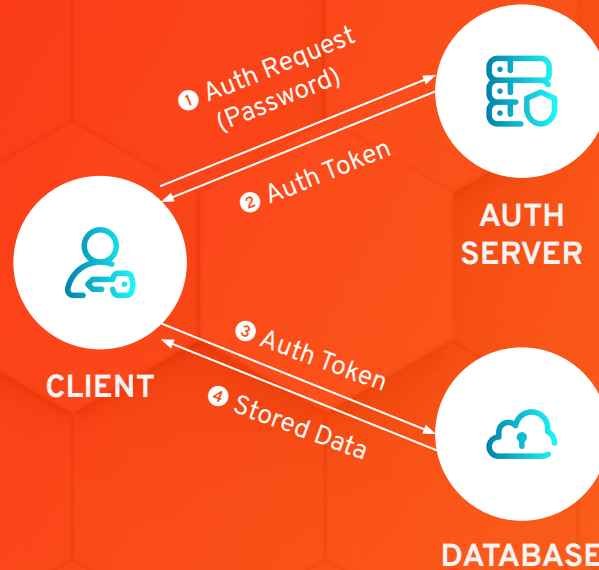
Vivek Nair

Ph.D. Student at UC Berkeley
<https://nair.me> · vivek@nair.me

Acknowledgments



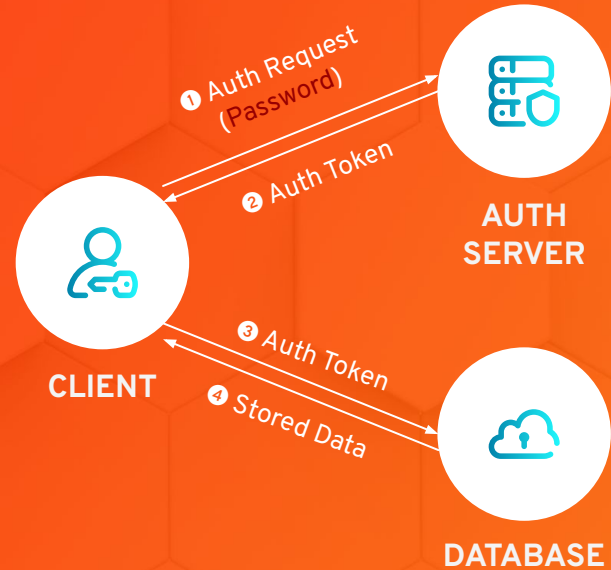
Password Management Service



Password Management Service

Two problems with this architecture:

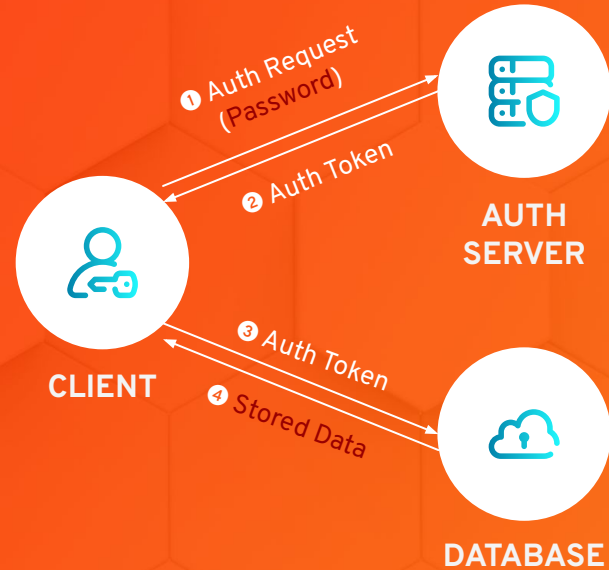
- Passwords are insecure



Password Management Service

Two problems with this architecture:

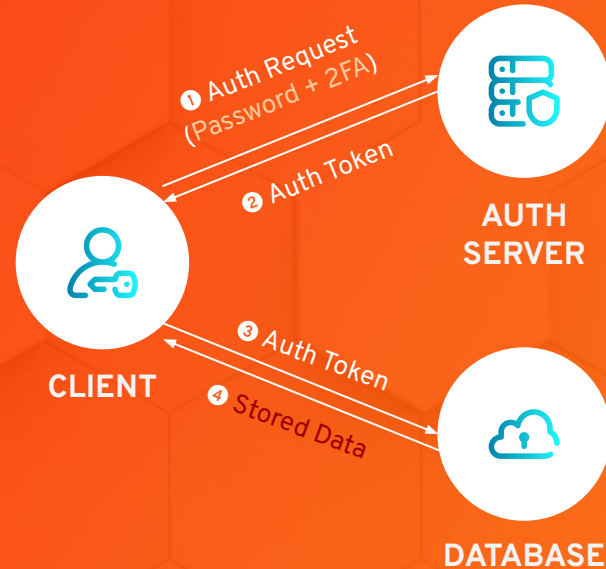
- Passwords are insecure
- Databases are leaky



Password Management Service

Two problems with this architecture:

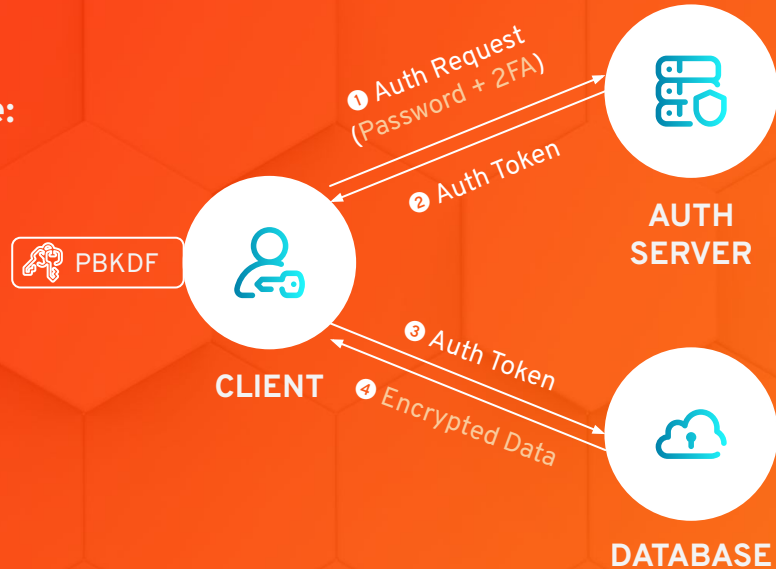
- Passwords are insecure
 - Add MFA!
- Databases are leaky



Password Management Service

Two problems with this architecture:

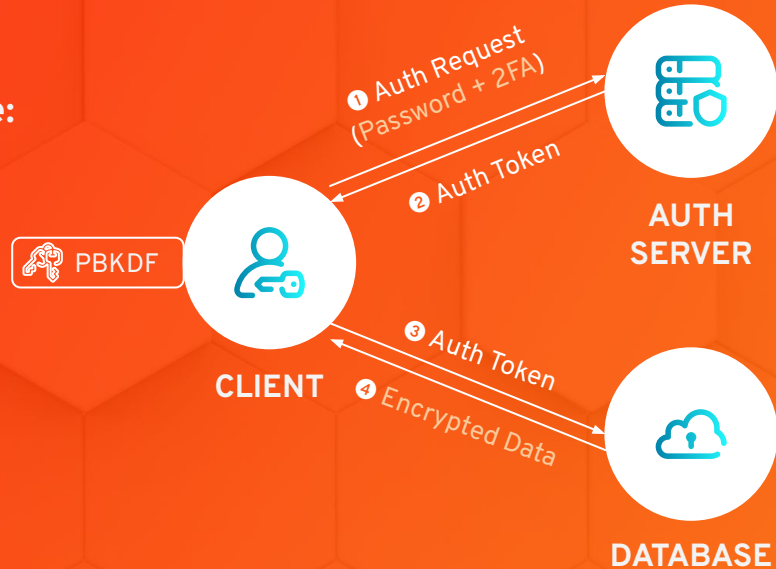
- Passwords are insecure
 - Add MFA!
- Databases are leaky
 - Add PBKDF!



Password Management Service

Two problems with this architecture:

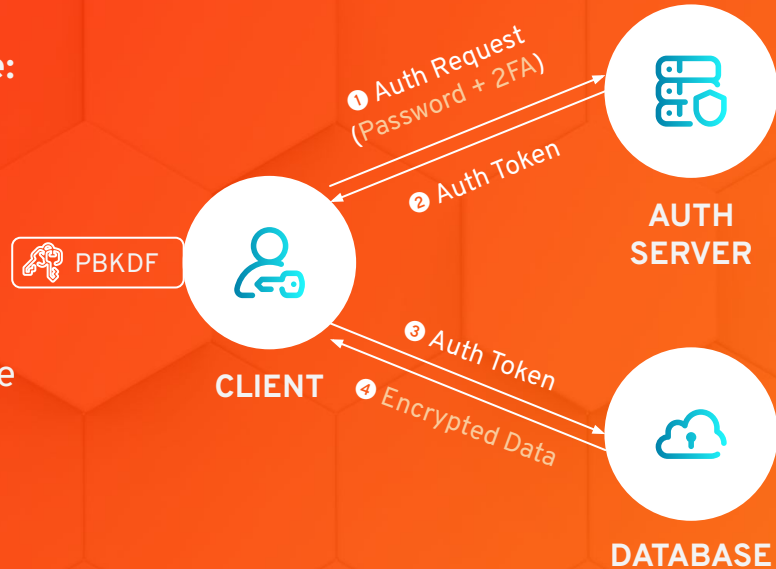
- Passwords are insecure
 - Add MFA!
- Databases are leaky
 - Add PBKDF!



Password Management Service

Two problems with this architecture:

- Passwords are insecure
 - Add MFA!
- Databases are leaky
 - Add PBKDF!
- Can we incorporate MFA into the key derivation function itself?



MULTI-FACTOR KEY DERIVATION

**MULTI-FACTOR
DERIVED KEY**



The MFKDF outputs a key as a function of all input factors



FACTOR 01

eg. a Password



FACTOR 02

eg. a TOTP Code



FACTOR 03

eg. a U2F Token



FACTOR 04

eg. Biometric Data



FACTOR 01

eg. a Password



hunter2

One-Way Function (OWF)



STATIC KEY

FACTOR 02

eg. a TOTP Code



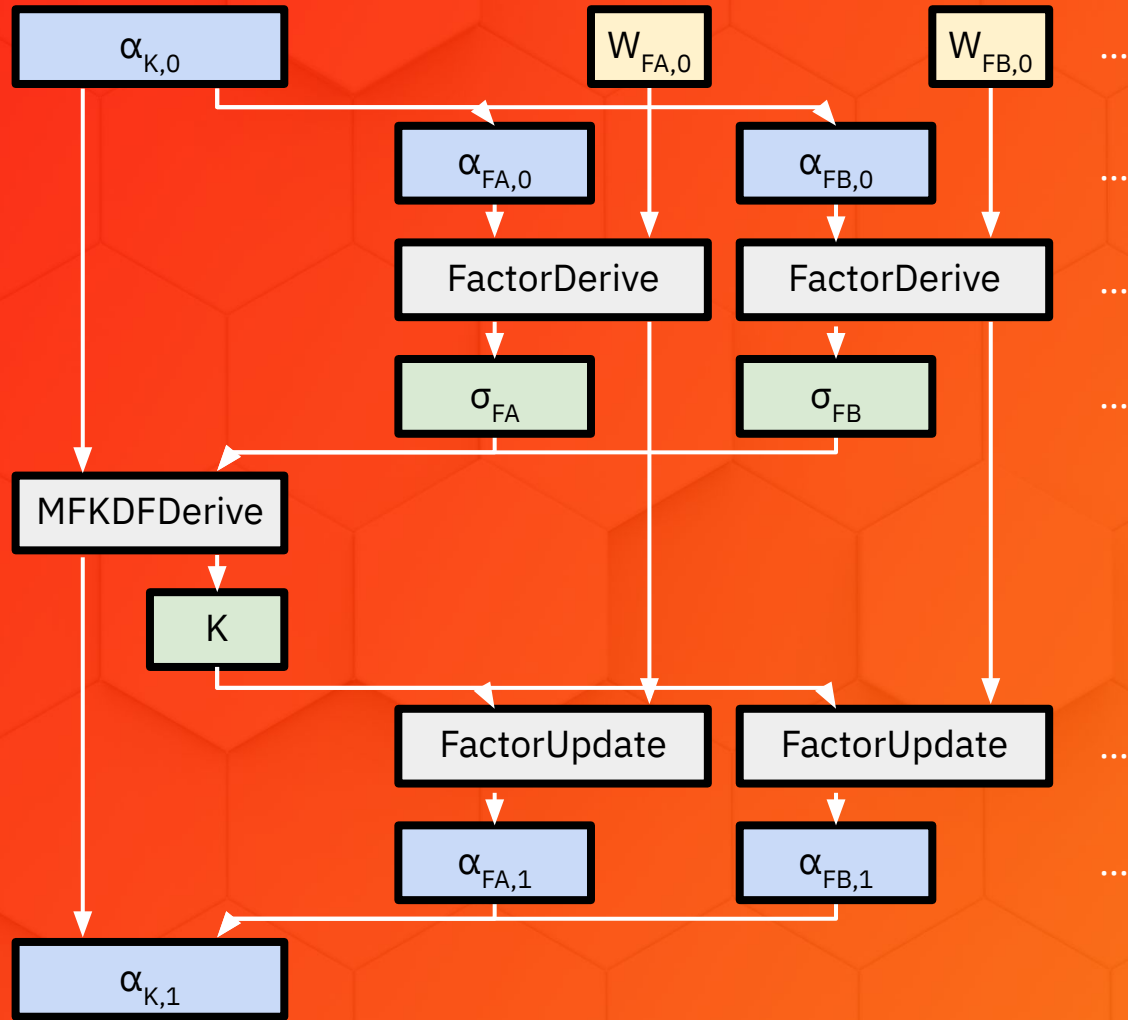
196353
778449
843812
234823
...

???

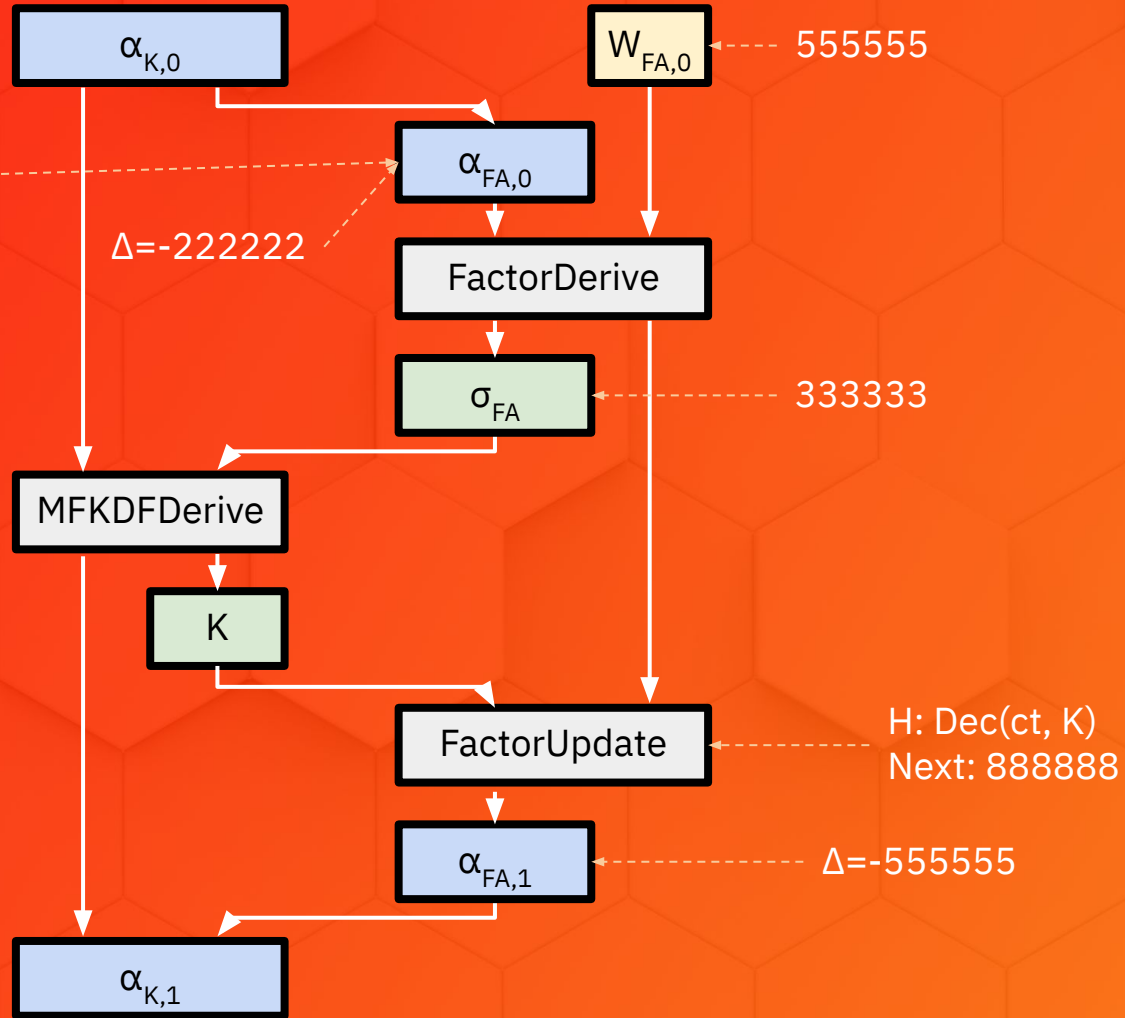


STATIC KEY

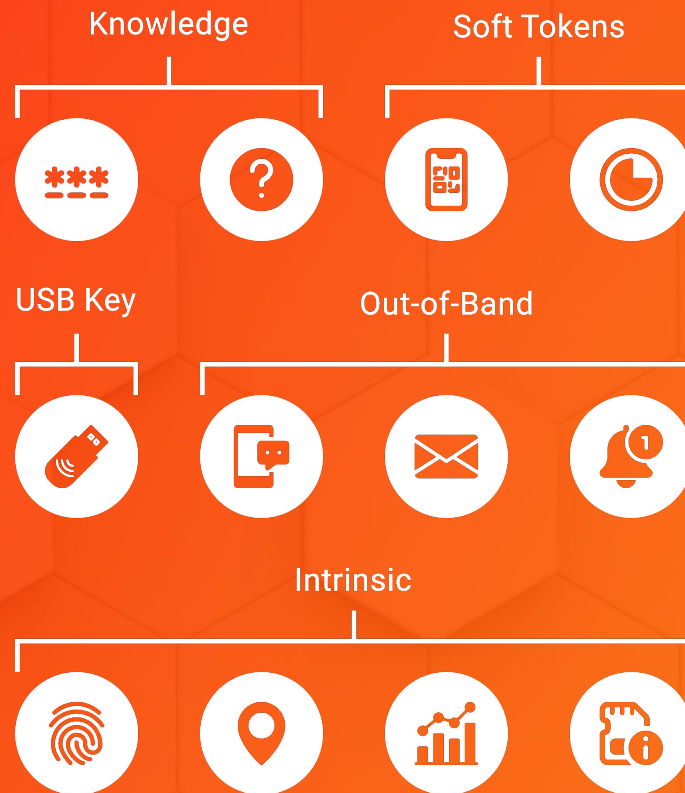
1st
derivation



K: Derived Key
H: HOTP Key
ct: Enc(H, K)



Other Factors:



Entropy & Brute Force



PBKDF

DK = PBKDF2(PRF, Password, Salt, Rounds, dkLen)

Intentionally inefficient!



MFKDF

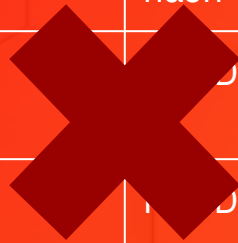
DK = MFAKDF(PRF, [f1,f2,...fn], Rounds, dkLen)
= PBKDF2(PRF, f1 · f2 · f3, Salt, Rounds, dkLen)

Difficulty is on top of all authentication factors!



users

user	hash
user1	MD5(password)
user2	MD5(password)



users

user	hash
user1	MFKDF(password, HOTP, ...)
user2	MFKDF(password, HOTP, ...)

Key
 $\approx 40b$



Password
 $\approx 40b$

$\approx 1 \text{ hour}$ → Key $\approx 40b$

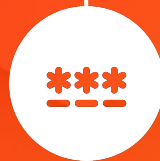


Password $\approx 40b$

Key $\approx 162b$ ← $\approx 10^{100} \text{ years}$



2/3



Password $\approx 40b$




YubiKey $160b$

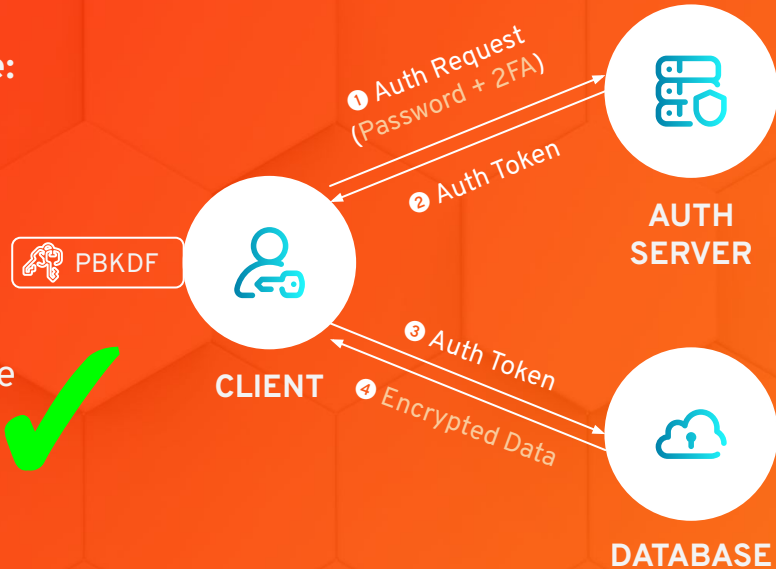


Code $122b$

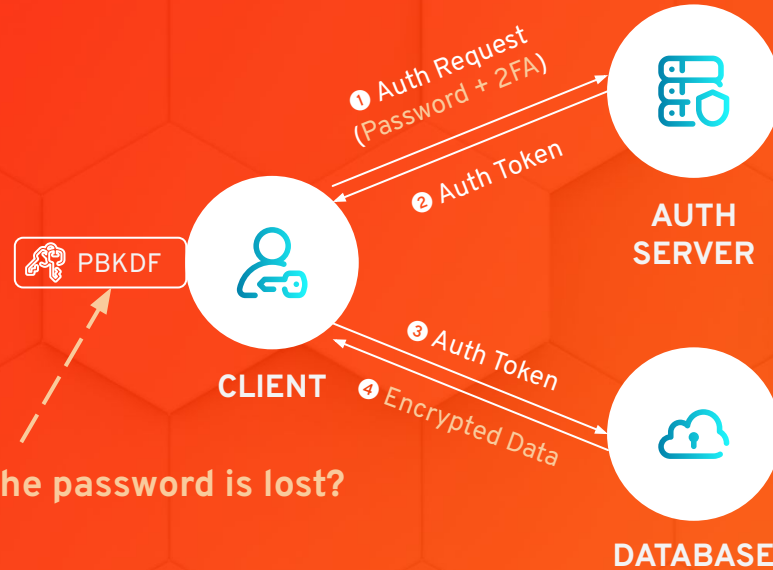
Password Management Service

Two problems with this architecture:

- Passwords are insecure
 - Add MFA!
- Databases are leaky
 - Add PBKDF!
- Can we incorporate MFA into the key derivation function itself? 



Password Management Service



What happens if the password is lost?

NIST SP 800-57: Key Recovery

Data Encryption Key (DEK): Used to encrypt user data

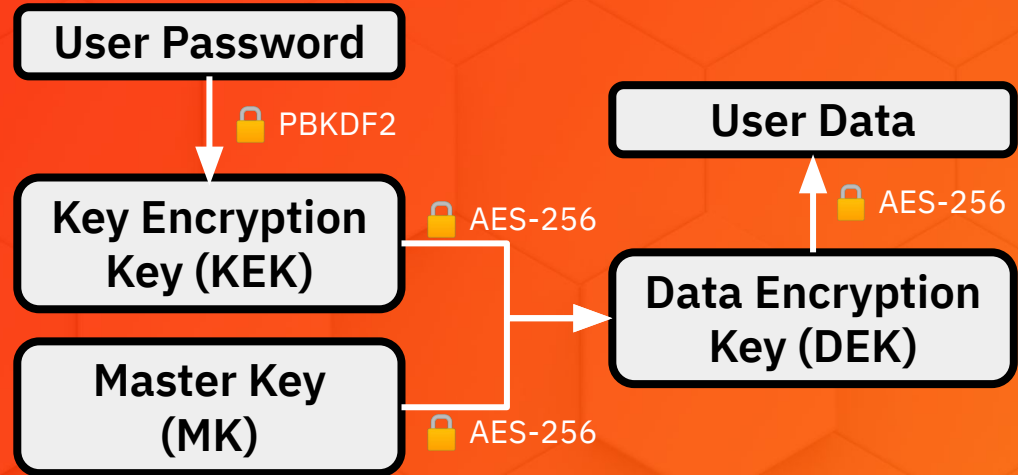
- $EncData = Enc(Data, DEK)$

Key Encryption Key (KEK): Password-derived key used to encrypt DEK

- $EncKey = Enc(DEK, KEK)$

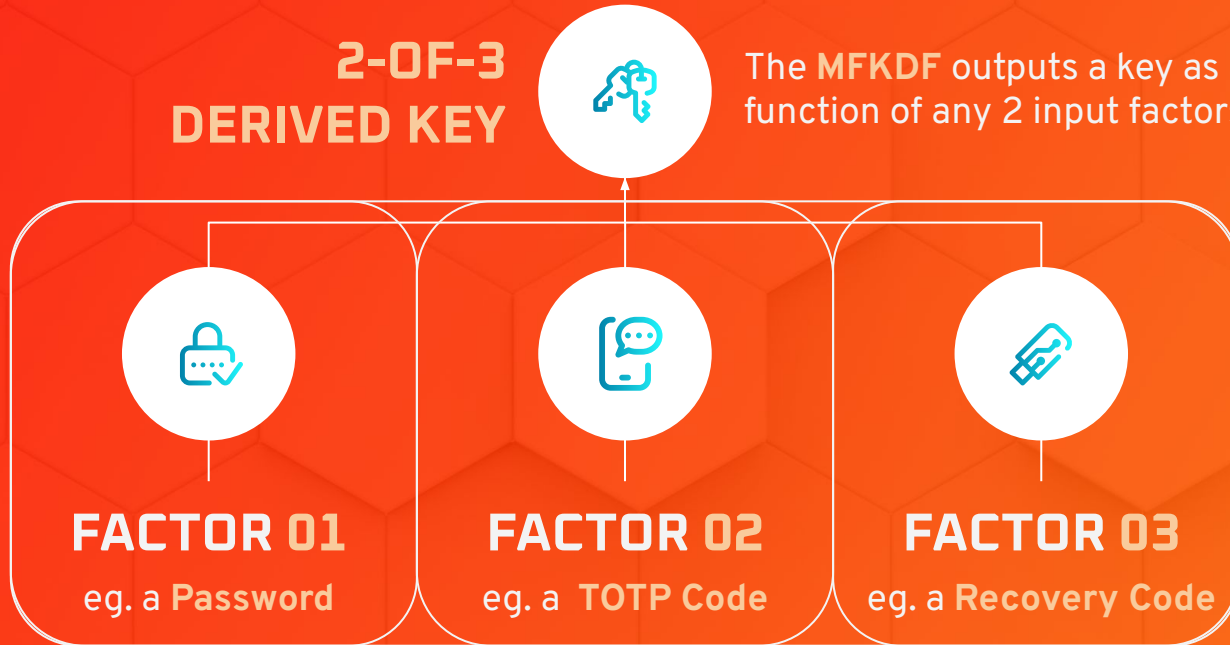
Master Key (MK): Centrally-stored key used to recover DEK

- $RecKey = Enc(DEK, MK)$



Single point of failure

THRESHOLD MULTI-FACTOR KEY DERIVATION



Key Stacking

**2-OF-3
DERIVED KEY**



The MFKDF outputs a key as a function of any 2 input factors



FACTOR 01

eg. a Password



FACTOR 02

eg. a TOTP Code



FACTOR 03

eg. a Recovery Code





Key

2/3



Password



TOTP

AND

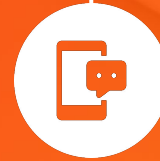


Email

OR

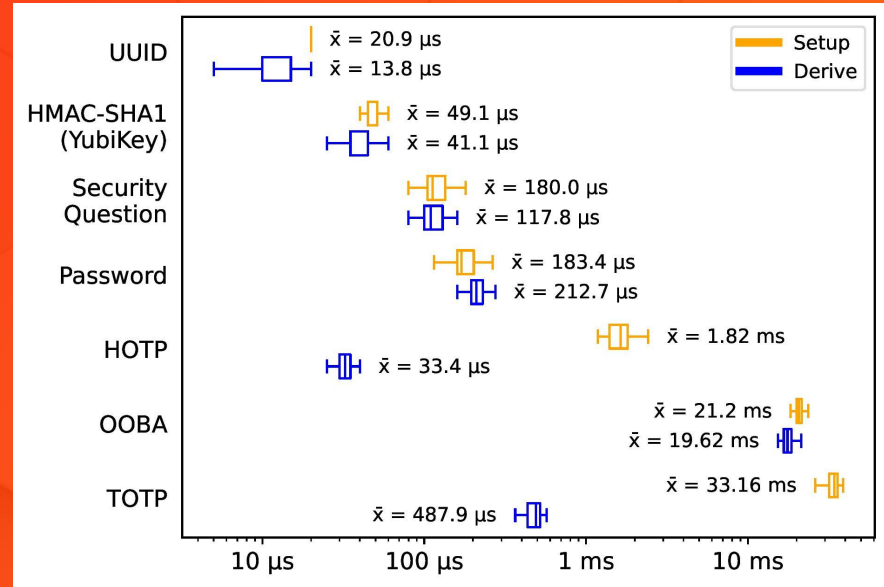
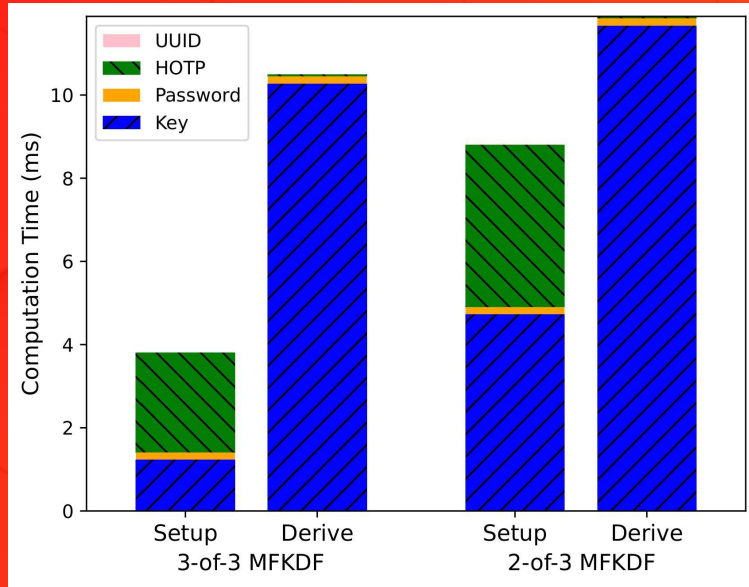


Security Questions



Recovery Code

Performance



mfkdf.com ← pbkdf2.com



Docs

Tutorials ▾

Testing

Coverage

Demos ▾

Videos

Get Started



Secure
based on argon2id



Fast
≤ 20ms overhead



Transparent
fully open-source



Flexible
modular design



Go beyond passwords

Most users have notoriously insecure passwords, with up to 81% of them re-using passwords across multiple accounts. MFKDF improves upon password-based key derivation by using all of a user's authentication factors (not just their password) to derive a key. MFKDF supports deriving key material from a variety of common factors, including HOTP, TOTP, and hardware tokens like YubiKey.

```
const derivedKey = await mfkdf.derive.key(JSON.parse(keyPolicy), {  
  password: mfkdf.derive.factors.password('Tr0ub4dour'),  
  hotp: mfkdf.derive.factors.hotp(365287),  
  recovery: mfkdf.derive.factors.uuid('9b1deb4d-3b7d-4bad-9bdd-2b0d7b3dcb6d')  
})  
  
console.log(derivedKey.key.toString('hex')) // -> 34d20ced439ec2f871c96ca377f25771
```

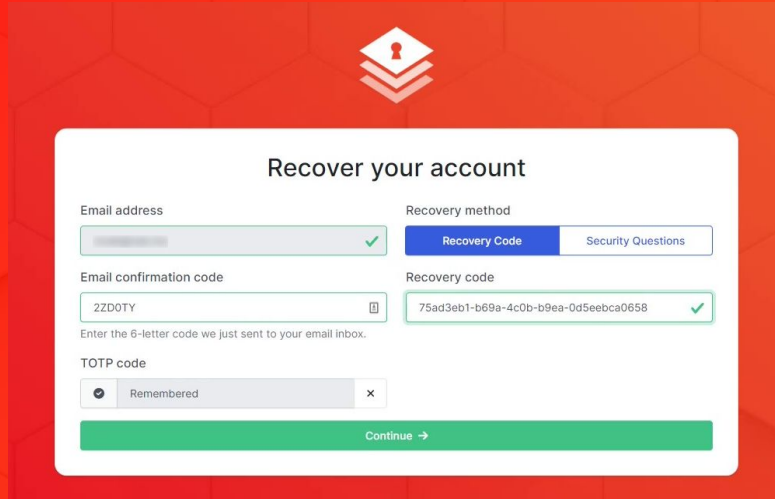
Increased key entropy

All factors must be simultaneously correctly guessed to derive a key using MFKDF, meaning that they can't be individually brute-force attacked. MFKDF keys are thus exponentially harder to crack while remaining just as fast to derive on the fly as password-derived keys for users with the correct credentials.



MFKDF

Centralized & Decentralized Demos



The image shows a 'Recover your account' form. At the top is the MFKDF logo. The form has two columns: 'Email address' and 'Recovery method'. The 'Email address' field contains a masked email and a green checkmark. The 'Recovery method' has two buttons: 'Recovery Code' (selected) and 'Security Questions'. Below are 'Email confirmation code' (2ZDOTY) and 'Recovery code' (75ad3eb1-b69a-4c0b-b9ea-0d5eebca0658) fields, both with green checkmarks. A 'TOTP code' section has a 'Remembered' checkbox. A green 'Continue' button is at the bottom.

Recover your account

Email address: [masked] ✓

Recovery method: **Recovery Code** | Security Questions

Email confirmation code: 2ZDOTY ✓

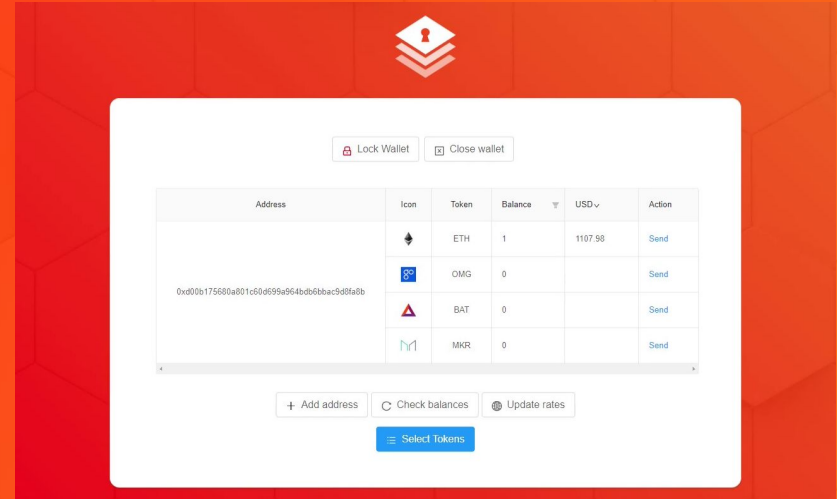
Recovery code: 75ad3eb1-b69a-4c0b-b9ea-0d5eebca0658 ✓

Enter the 6-letter code we just sent to your email inbox.

TOTP code: Remembered

Continue →

<https://demo.mfkdf.com>



The image shows a wallet interface. At the top is the MFKDF logo. There are 'Lock Wallet' and 'Close wallet' buttons. Below is a table of tokens. At the bottom are buttons for '+ Add address', 'Check balances', 'Update rates', and 'Select Tokens'.

Address	Icon	Token	Balance	USD	Action
	ETH icon	ETH	1	1107.58	Send
0xd00b17568fa801c68d959a964bd66bbac9d8a8b	OMG icon	OMG	0		Send
	BAT icon	BAT	0		Send
	MKR icon	MKR	0		Send

+ Add address | Check balances | Update rates

Select Tokens

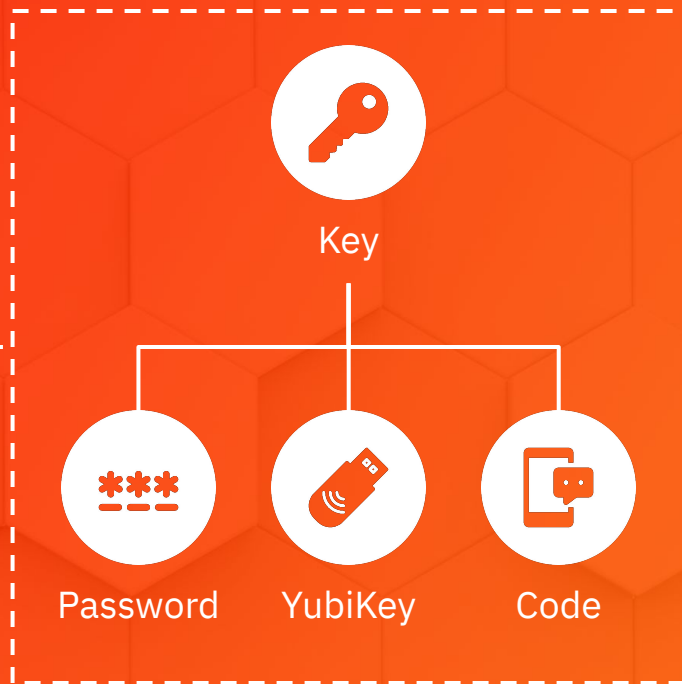
<https://wallet.mfkdf.com>



IPNS
eigb...oaw



IPFS
bafk...vx3u



MFKDF
Policy



MFKDF

Custodial Wallet

✓ Portability

✓ Recoverability

✓ MFA

✓ Common Factors

✗ Decentralized

✗ Trustless

Non-custodial Wallet

✗ Portability

✗ Recoverability

✗ MFA

✗ Common Factors

✓ Decentralized

✓ Trustless

MFKDF Wallet

✓ Portability

✓ Recoverability

✓ MFA

✓ Common Factors

✓ Decentralized

✓ Trustless

PBKDF2 is also used in...



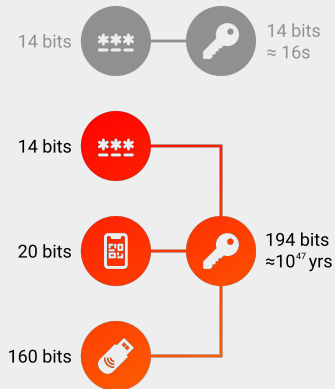
MFKDF Summary



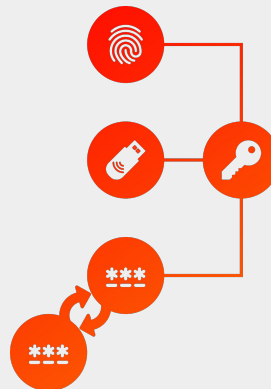
USABILITY & FACTOR COMPATIBILITY



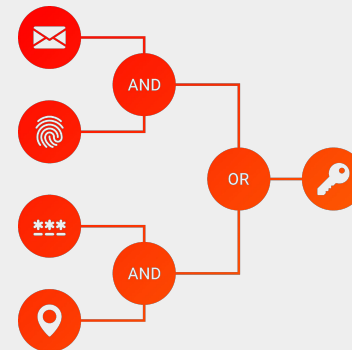
EXPONENTIAL SECURITY



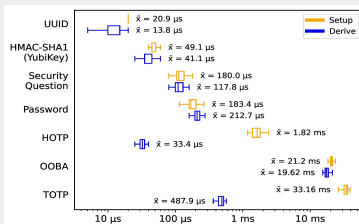
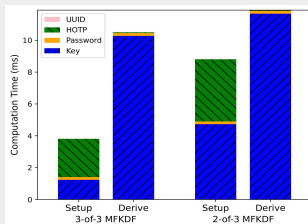
CLIENT-SIDE RECOVERY



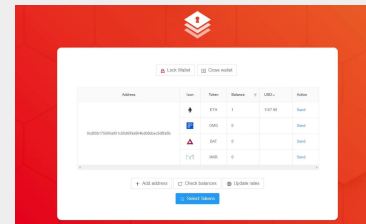
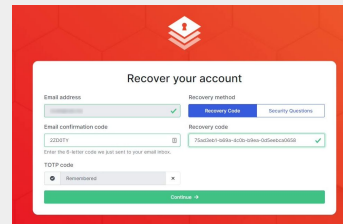
POLICY ENFORCEMENT



HIGHLY PERFORMANT



NEW & EXISTING APPLICATIONS





Thanks!



<https://mfkdf.com>



<https://arxiv.org/abs/2208.05586>



<https://github.com/multifactor/mfkdf>