# Multi-Factor Key Derivation Function (MFKDF)

# Vivek Nair

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

# Acknowledgments

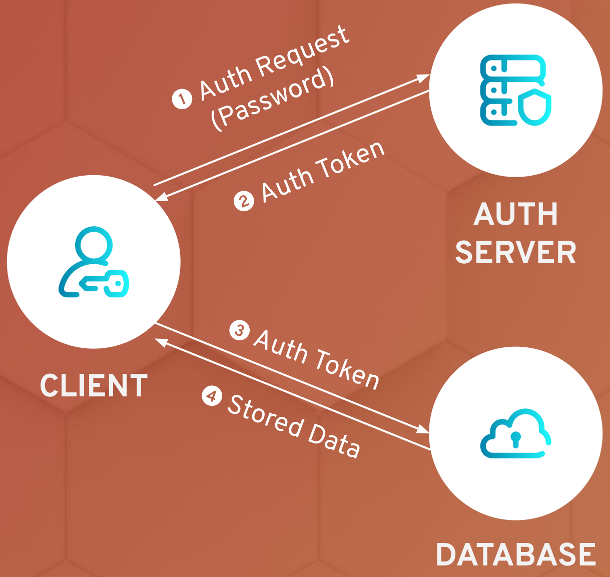
NSF

National Physical Science Consortium

the Hertz FOUNDATION

Berkeley RDI

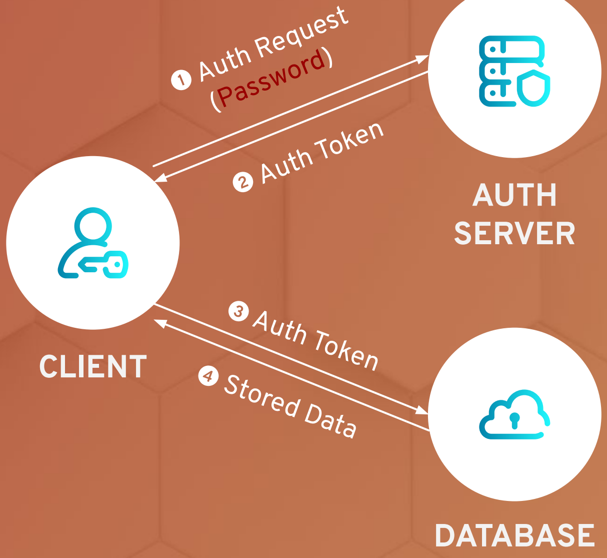
# Password Management Service



# Password Management Service

**Two problems with this architecture:**

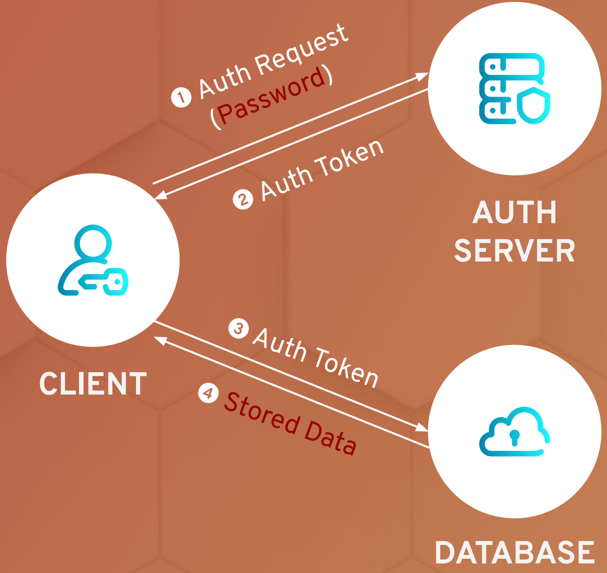
- Passwords are insecure



# Password Management Service

Two problems with this architecture:

- Passwords are insecure
- Databases are leaky

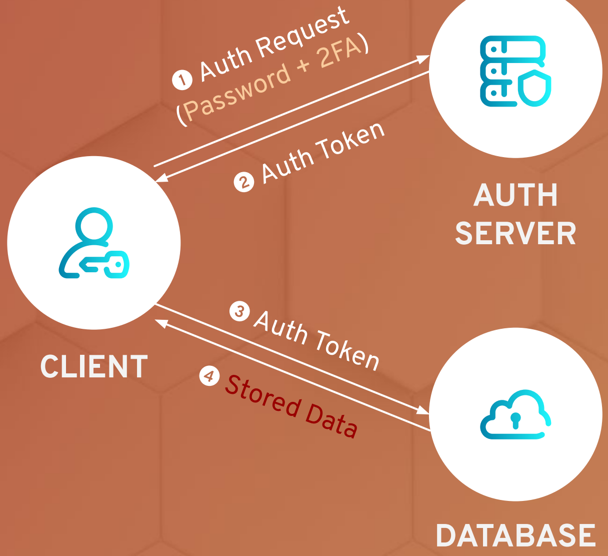


MFKDF

# Password Management Service

Two problems with this architecture:

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

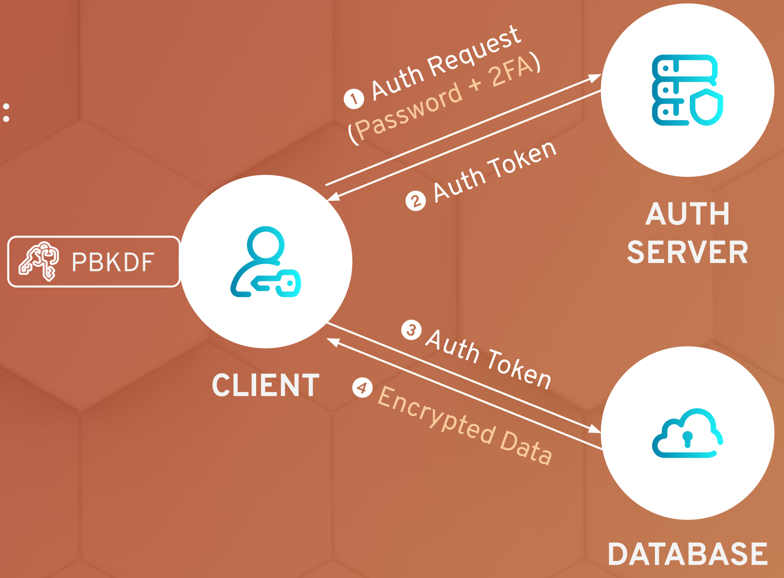


MFKDF

# Password Management Service

**Two problems with this architecture:**

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

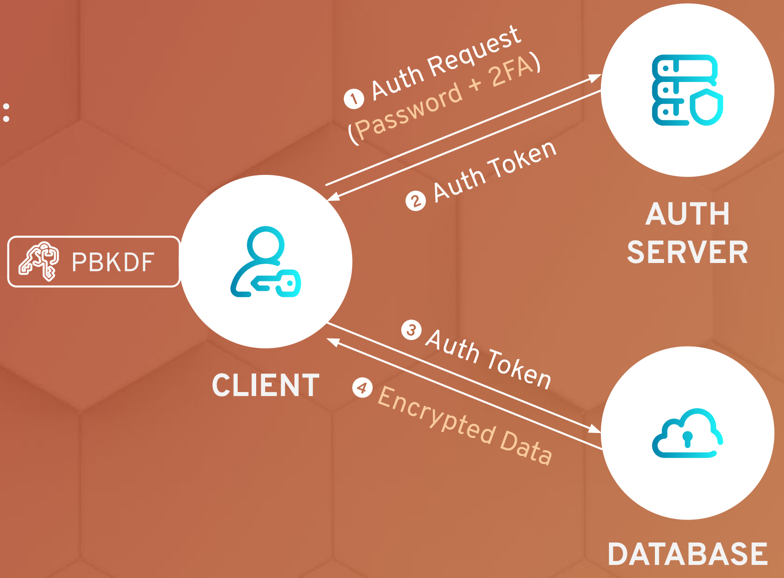


MFKDF

# Password Management Service

**Two problems with this architecture:**

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

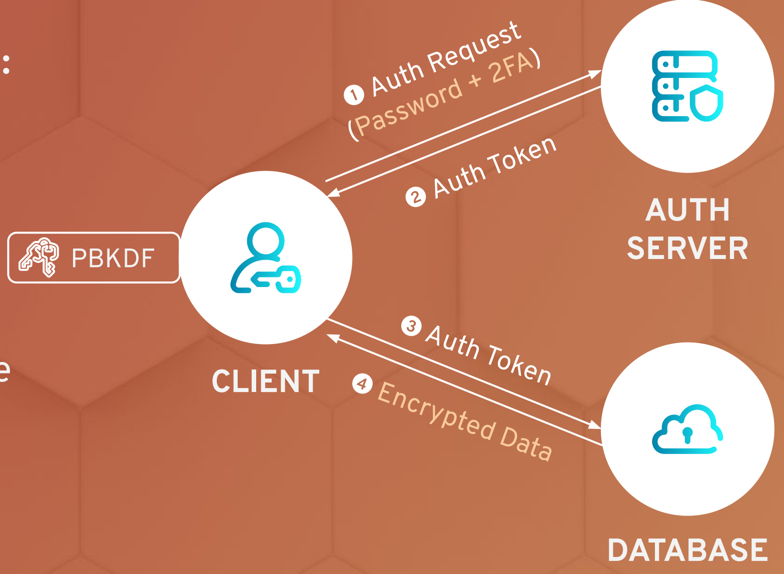


# Password Management Service

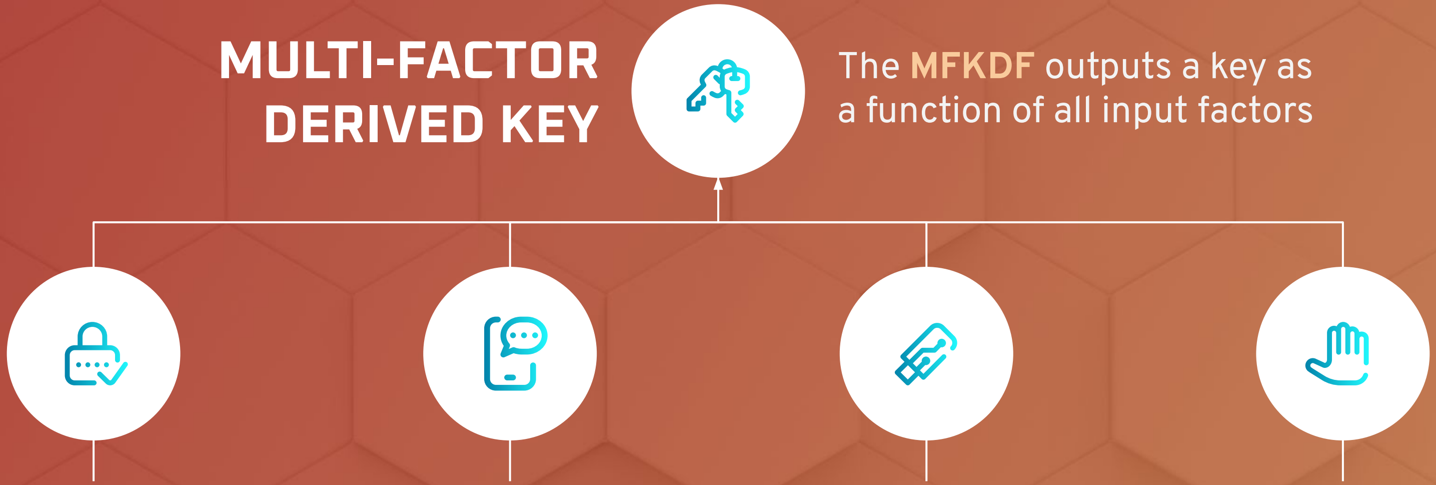
**Two problems with this architecture:**

- 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**

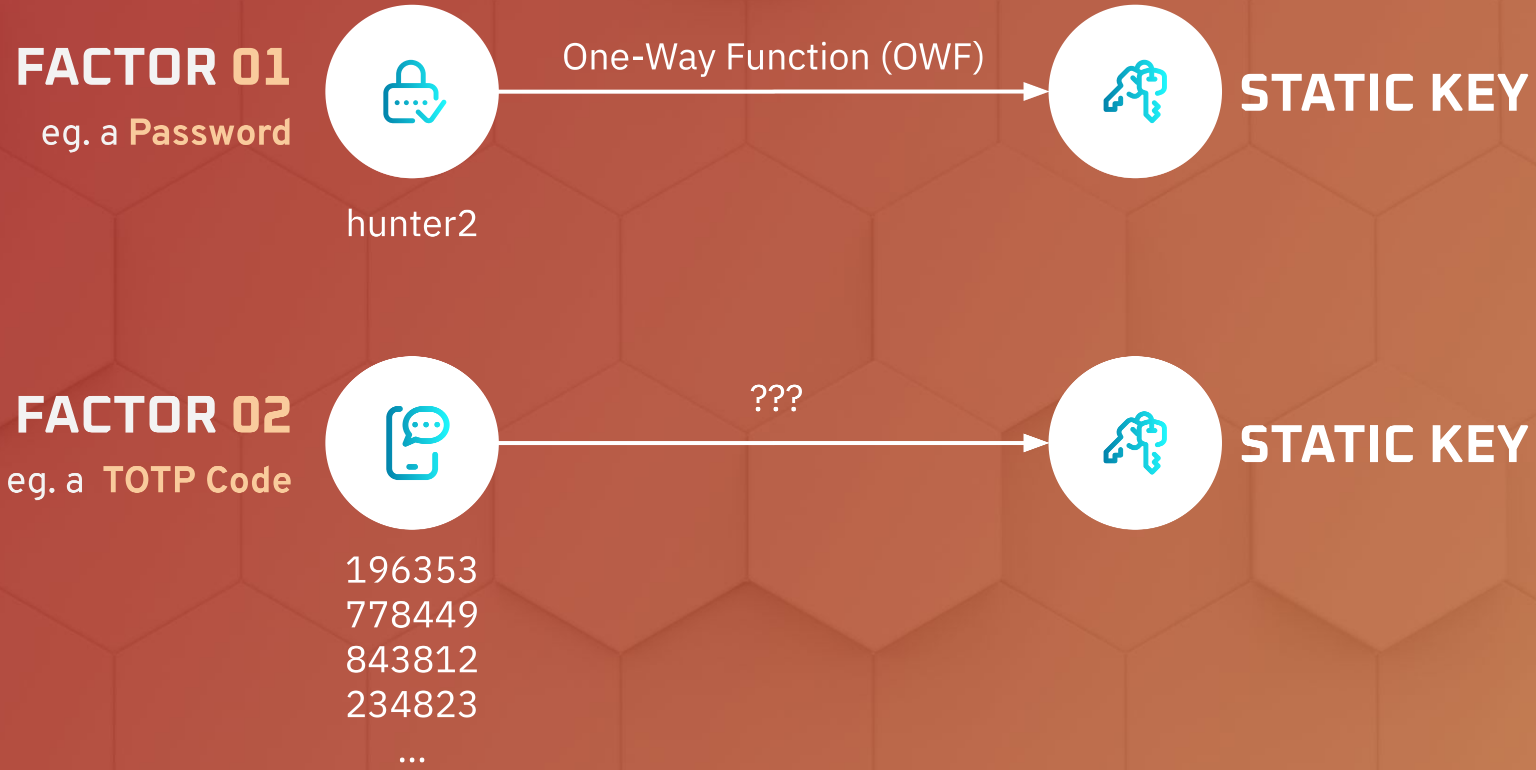
The MFKDF outputs a key as a function of all input factors

**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**

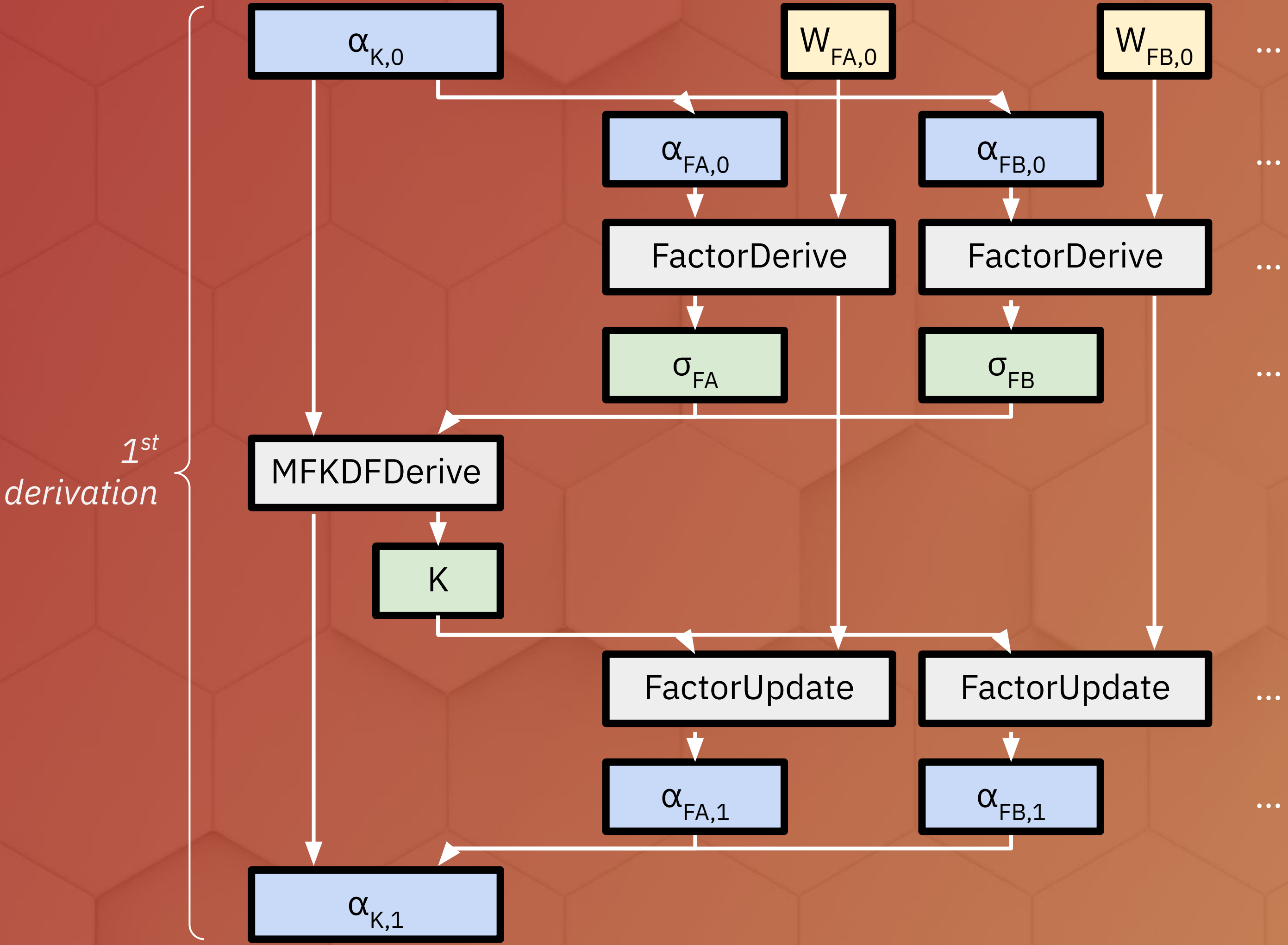
**FACTOR 02**

eg. a TOTP Code

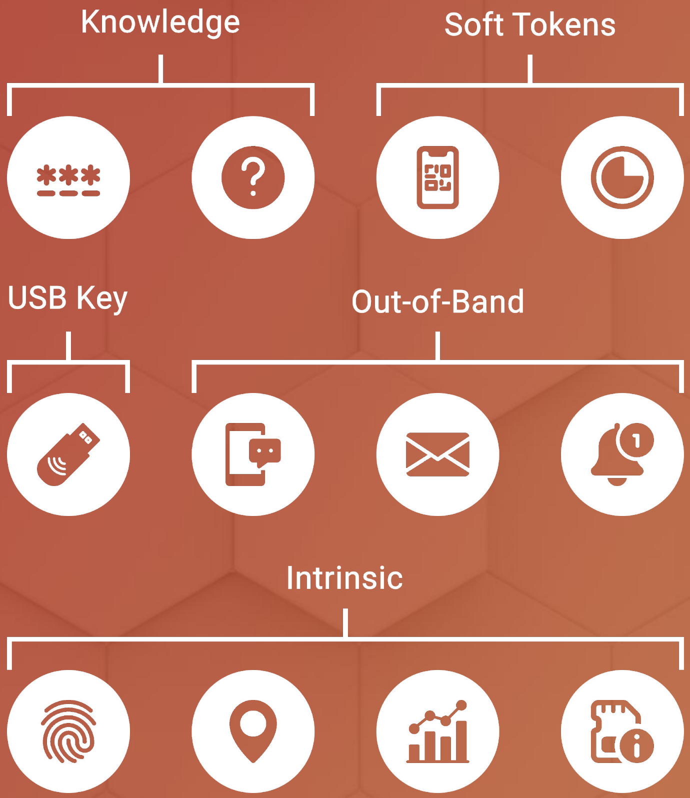
196353
778449
843812
234823
...

???

**STATIC KEY**


$\alpha_{K,0}$
$W_{FA,0}$
$W_{FB,0}$
...
$\alpha_{FA,0}$
$\alpha_{FB,0}$
...
FactorDerive
FactorDerive
...
$\sigma_{FA}$
$\sigma_{FB}$
...
$1^{st}$ derivation
MFKDFDerive
K
FactorUpdate
FactorUpdate
...
$\alpha_{FA,1}$
$\alpha_{FB,1}$
...
$\alpha_{K,1}$

MFKDF


Knowledge
Soft Tokens
USB Key
Out-of-Band
Intrinsic

# Entropy & Brute Force

Intentionally inefficient!

## PBKDF

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

## MFKDF

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

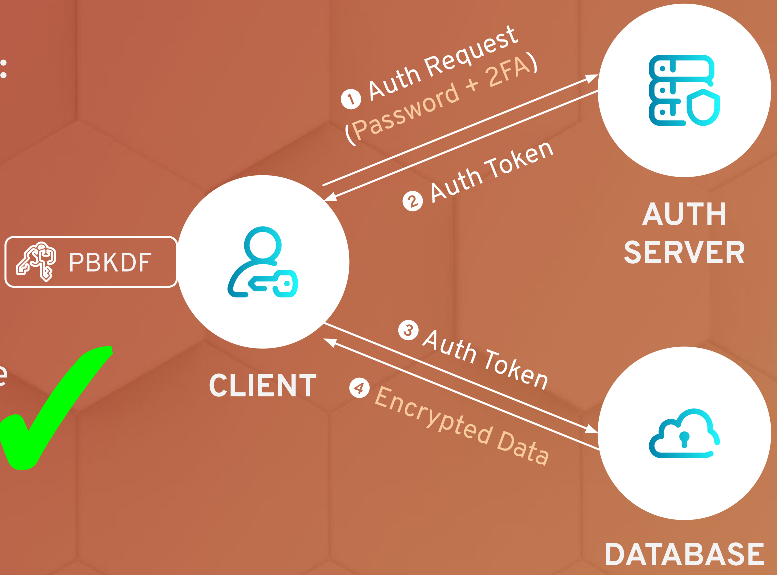
Difficulty is on top of all
authentication factors!

# Password Management Service

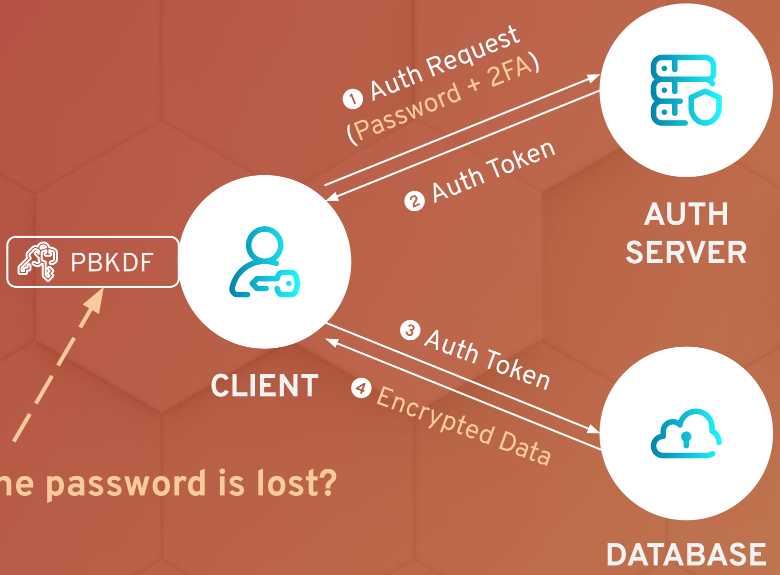
Two problems with this architecture:

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

- Can we incorporate MFA into the key derivation function itself? ✓

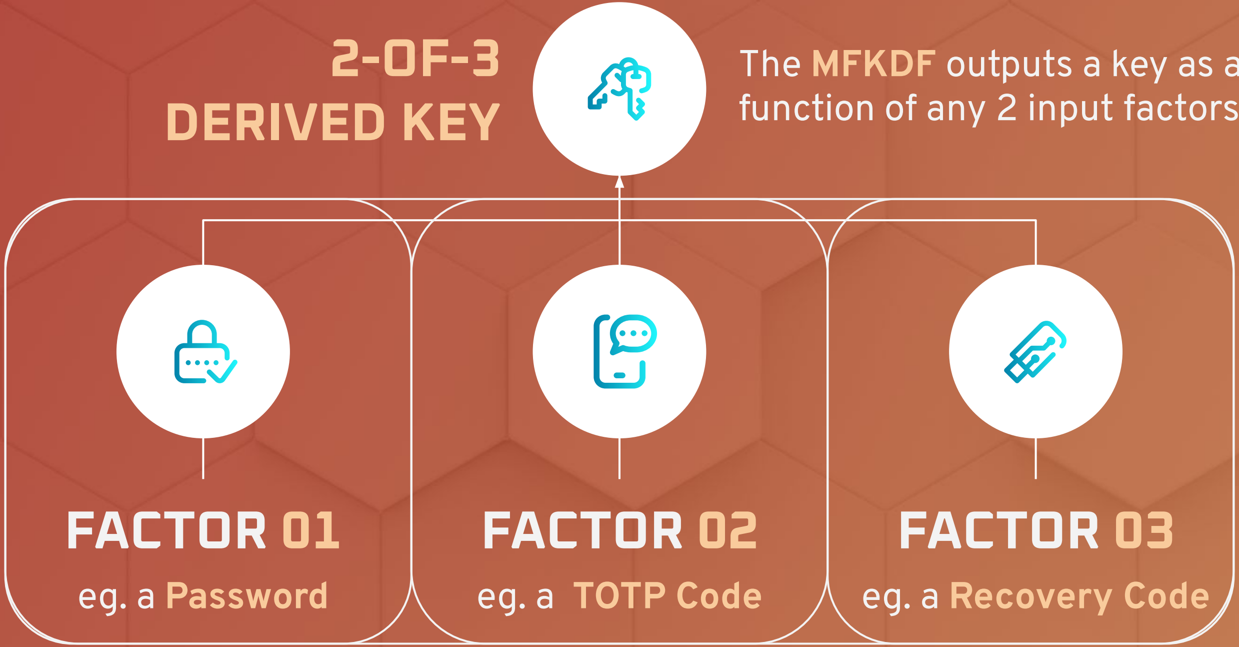


# Password Management Service



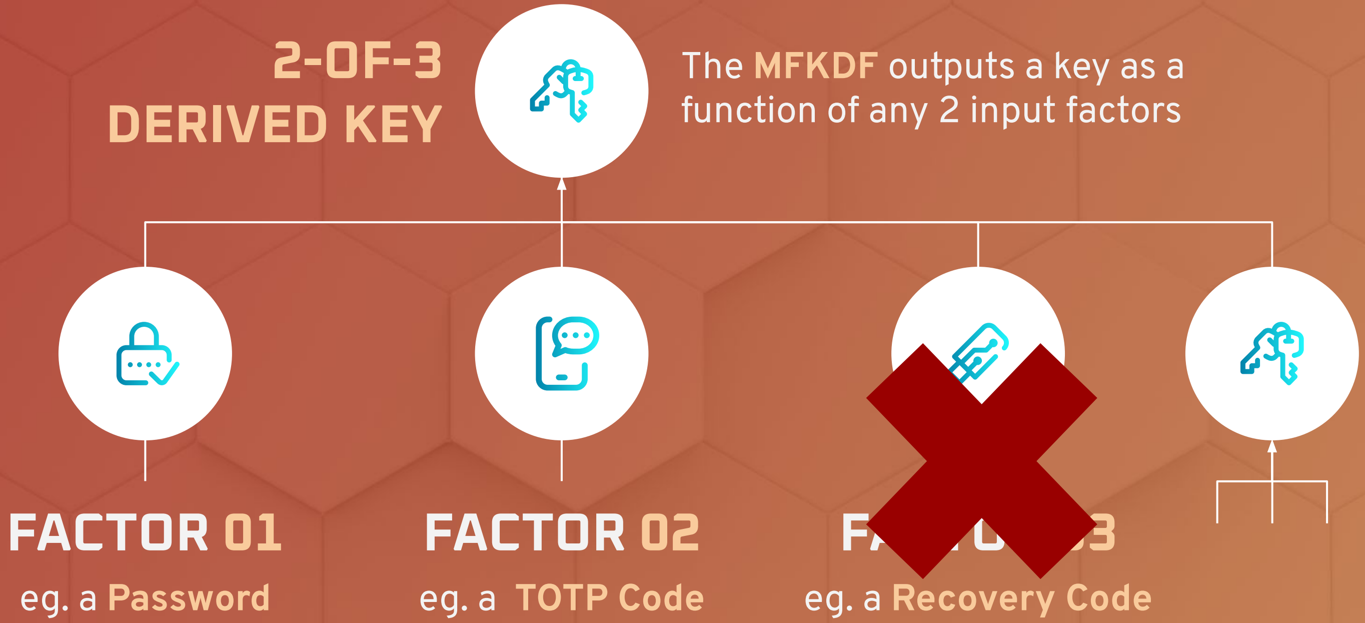
What happens if the password is lost?

# THRESHOLD MULTI-FACTOR KEY DERIVATION



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

# Key Stacking



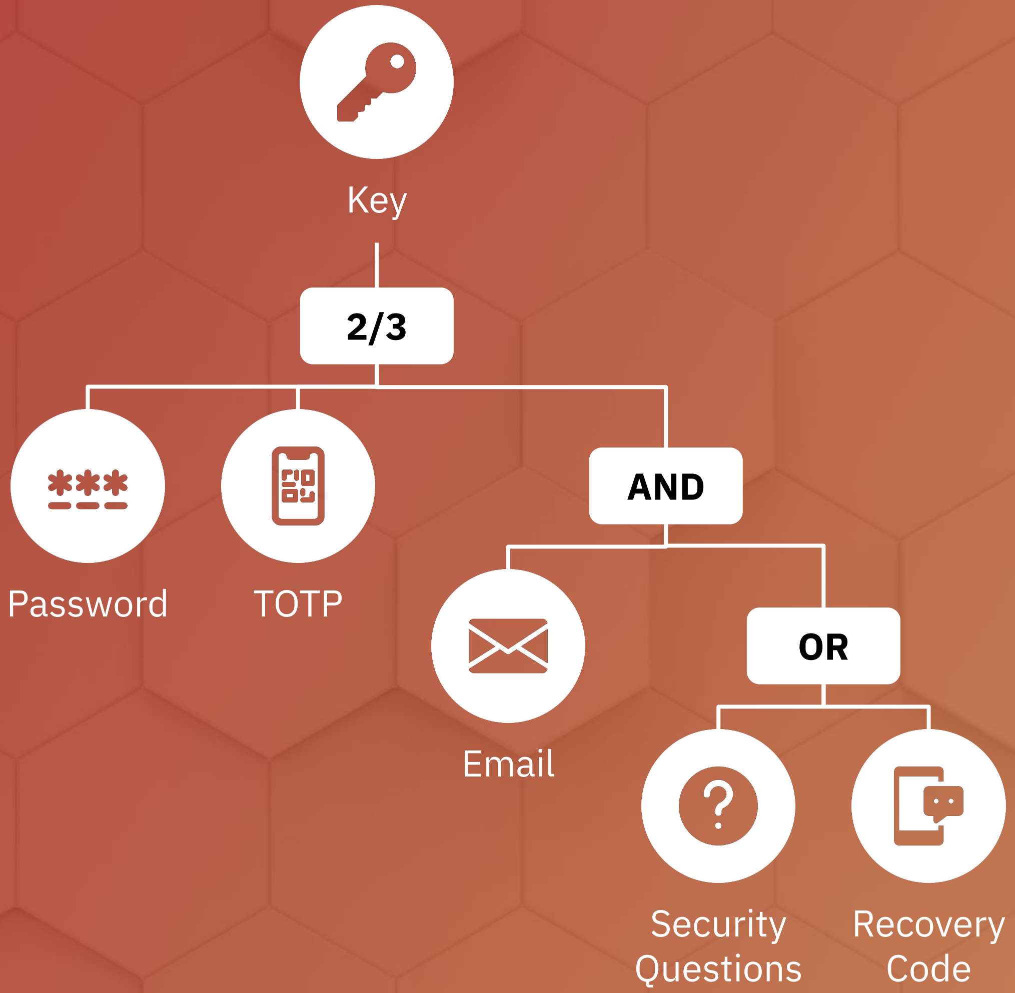
2-OF-3
DERIVED KEY

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

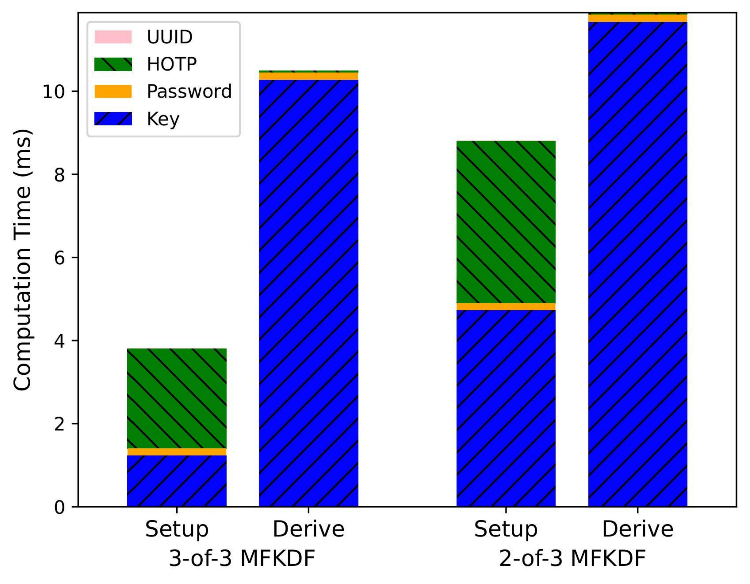
Email

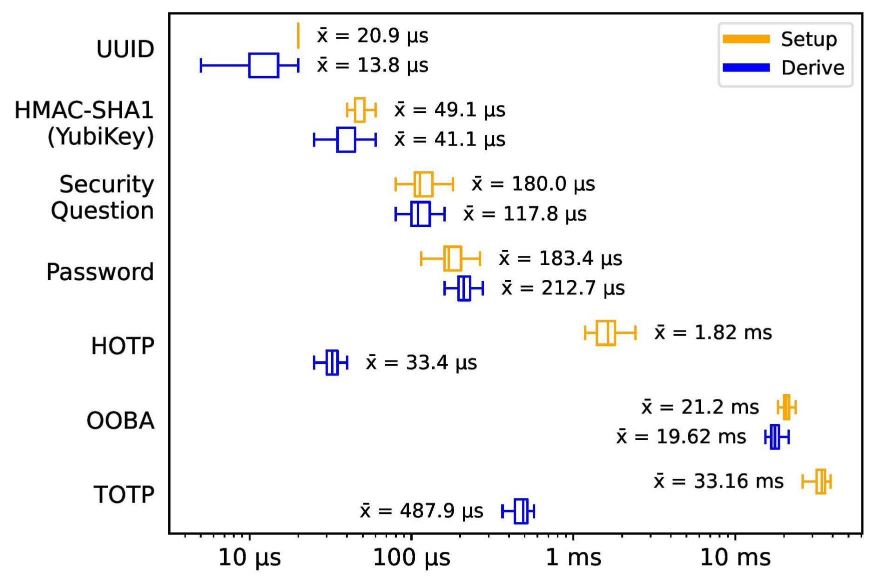
OR

Security Questions

Recovery Code

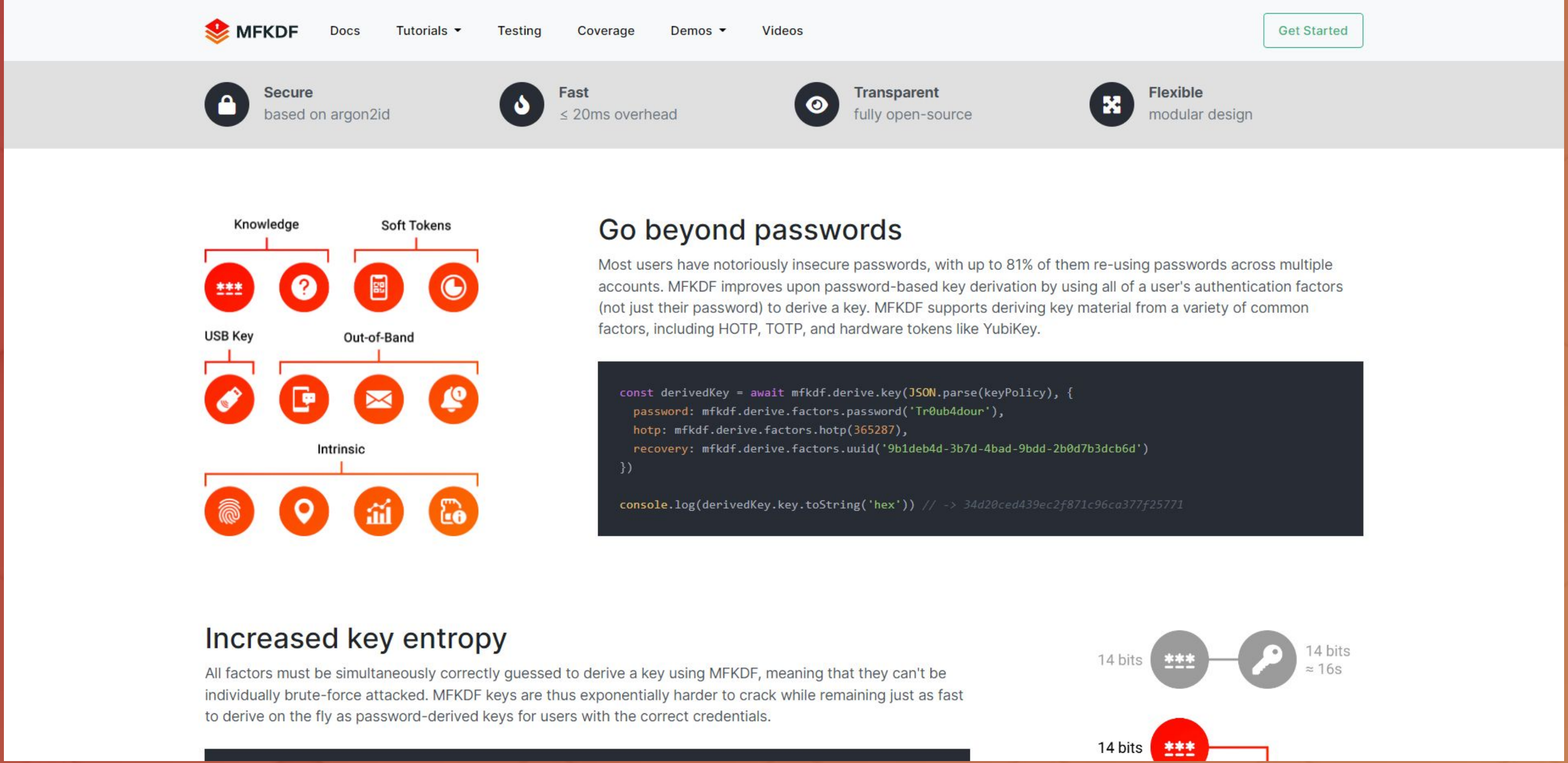
# Performance





MFKDF

# mfkdf.com ← pbkdf2.com



**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.

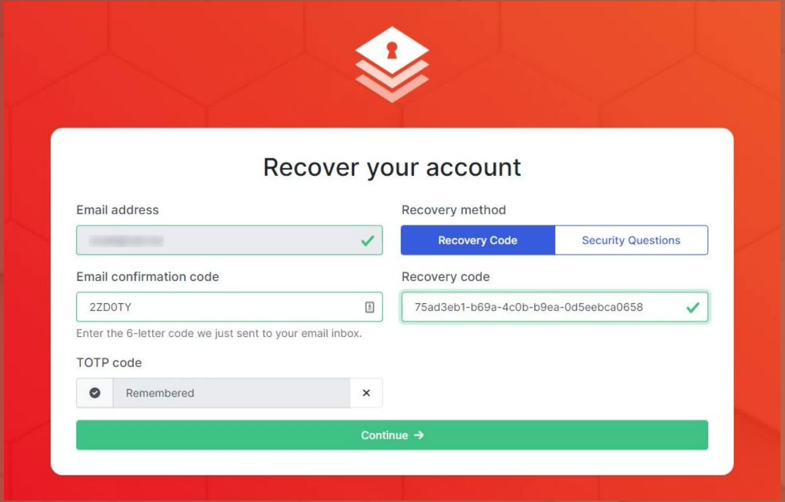
```
const derivedKey = await mfkdf.derive.key(JSON.parse(keyPolicy), {
  password: mfkdf.derive.factors.password('Tr0ub4dour'),
  hotp: mfkdf.derive.factors.hotp(365287),
  recovery: mfkdf.derive.factors.uuid('9b1deb4d-3b7d-4bad-9bdd-2b0d7b3dcb6d')
})

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.

# Centralized & Decentralized Demos



Recover your account

Email address: [redacted] ✓

Recovery method: ● Recovery Code ○ Security Questions

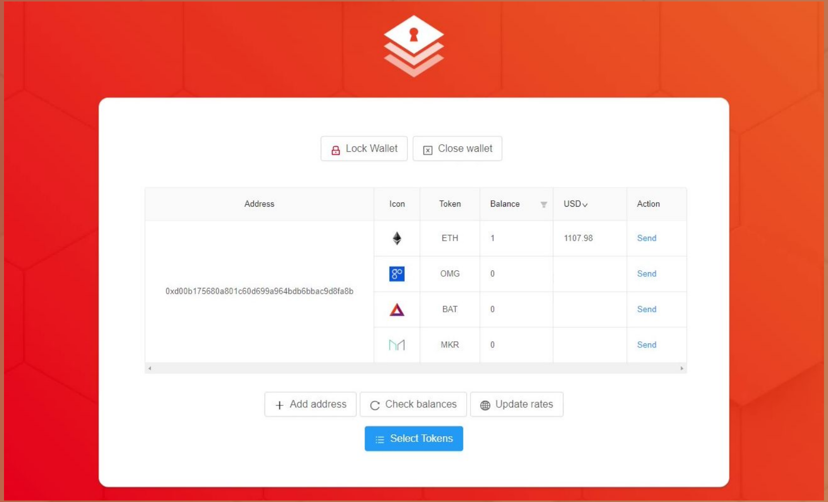
Email confirmation code: 2ZD0TY

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

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

TOTP code: ✓ Remembered ×

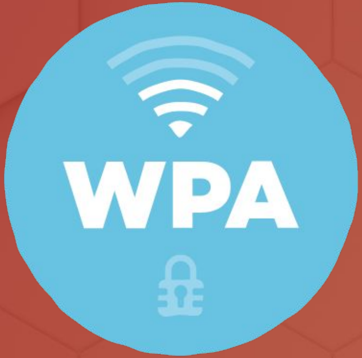
https://demo.mfkdf.com

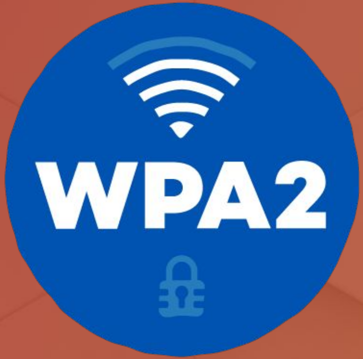


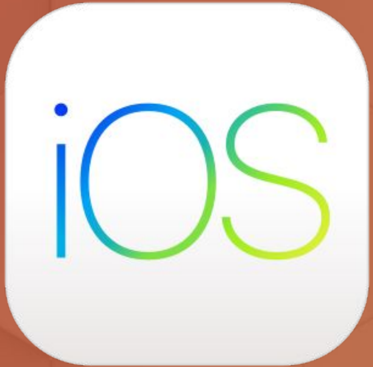
| Address | Icon | Token | Balance | USD | Action |
|---|---|---|---|---|---|
| 0xd00b175680a801c60d699a964bdb6bbac9d8fa8b |  | ETH | 1 | 1107.98 | Send |
|  |  | OMG | 0 |  | Send |
|  |  | BAT | 0 |  | Send |
|  |  | MKR | 0 |  | Send |

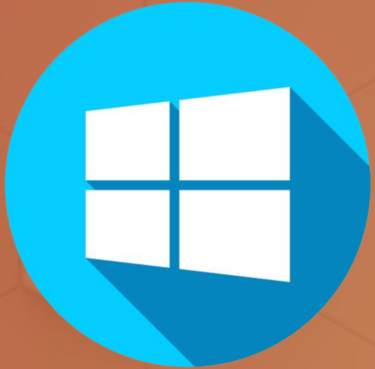
https://wallet.mfkdf.com

# PBKDF2 is also used in...





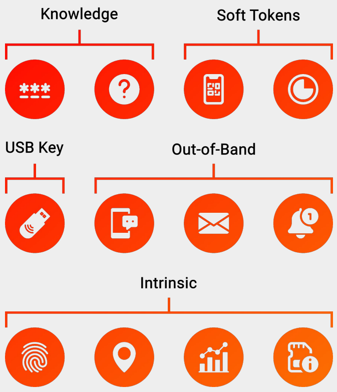




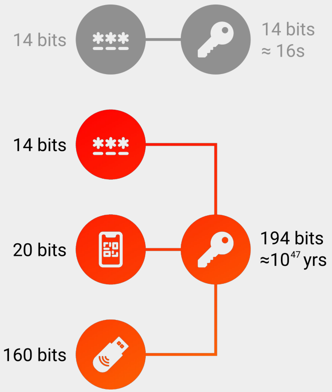
# MFKDF Summary

MFKDF

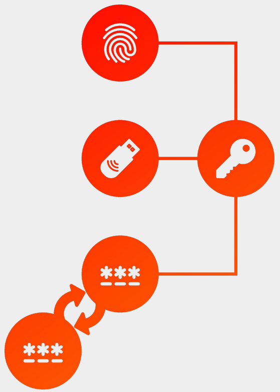
## USABILITY & FACTOR COMPATIBILITY



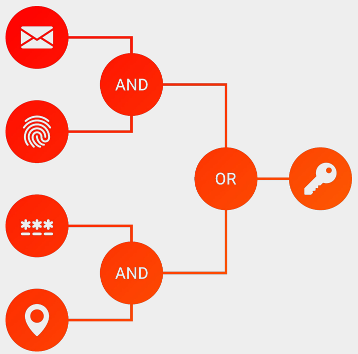
## EXPONENTIAL SECURITY



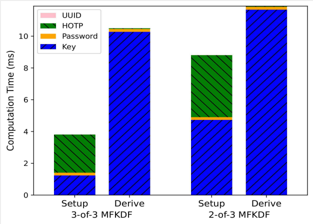
## CLIENT-SIDE RECOVERY

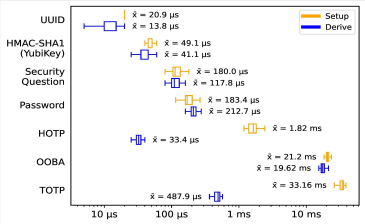


## POLICY ENFORCEMENT

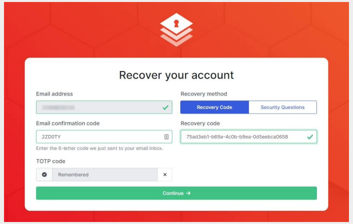


## HIGHLY PERFORMANT





## NEW & EXISTING APPLICATIONS



Recover your account

Email address: [redacted] ✓

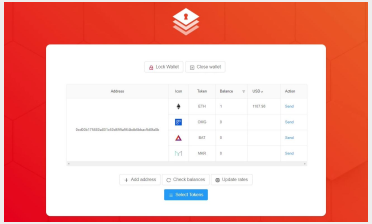
Recovery method: **Recovery Code** | Security Questions

Email confirmation code: 2ZD0TY

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

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

TOTP code: ● Remembered ×



| Address | Icon | Token | Balance | USD | Action |
|---|---|---|---|---|---|
| 0xd00b175680a801c60d699a964bdb6bbac9d8fa8b | | ETH | 1 | 1107.98 | Send |
| | | OMG | 0 | | Send |
| | | BAT | 0 | | Send |
| | | MKR | 0 | | Send |

# Thanks!

https://mfkdf.com

https://arxiv.org/abs/2208.05586

https://github.com/multifactor/mfkdf