Multi-Factor Key Derivation Function (MFKDF)
Vivek Nair

Ph.D. Student at UC Berkeley
https://nair.me · vivek@nair.me
Acknowledgments
Password Management Service

1. Auth Request (Password)
2. Auth Token
3. Auth Token

CLIENT

AUTH SERVER

DATABASE

MFKDF
Two problems with this architecture:

- Passwords are insecure
Password Management Service

Two problems with this architecture:

- Passwords are insecure
- Databases are leaky
Two problems with this architecture:

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

---

**FACTOR 02**

*eg. a TOTP Code*

 196353
 778449
 843812
 234823
 ...

---

One-Way Function (OWF)

---

**STATIC KEY**

---

**STATIC KEY**
\[ \alpha_{K,0} \quad \text{FactorDerive} \quad \alpha_{FB,0} \]

\[ \alpha_{FA,0} \quad \sigma_{FA} \quad \alpha_{FB,0} \quad \sigma_{FB} \quad \text{FactorUpdate} \quad \alpha_{FB,1} \]

\[ \alpha_{FA,1} \quad \text{FactorUpdate} \quad \alpha_{FB,1} \]

\[ \alpha_{K,1} \]

\[ 1^{st} \text{ derivation} \]
Entropy & Brute Force

PBKDF

\[ \text{DK} = \text{PBKDF2} ( \text{PRF}, \text{Password}, \text{Salt}, \text{Rounds}, \text{dkLen}) \]

MFKDF

\[ \text{DK} = \text{MFAKDF} ( \text{PRF}, [f_1, f_2, ..., f_n], \text{Rounds}, \text{dkLen}) \]
\[ = \text{PBKDF2} ( \text{PRF}, f_1 \cdot f_2 \cdot f_3, \text{Salt}, \text{Rounds}, \text{dkLen}) \]

Intentionally inefficient!

Difficulty is on top of all authentication factors!
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

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 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
Performance

![Graph showing performance comparison of different methods](image)

- **UUID**: $\hat{x} = 20.9 \mu s$, $\bar{x} = 13.8 \mu s$
- **HMAC-SHA1 (YubiKey)**: $\hat{x} = 49.1 \mu s$, $\bar{x} = 41.1 \mu s$
- **Security Question**: $\hat{x} = 180.0 \mu s$, $\bar{x} = 117.8 \mu s$
- **Password**: $\hat{x} = 183.4 \mu s$, $\bar{x} = 212.7 \mu s$
- **HOTP**: $\hat{x} = 33.4 \mu s$
- **OOBA**: $\hat{x} = 21.2 \mu s$, $\bar{x} = 19.62 \mu s$
- **TOTP**: $\hat{x} = 487.9 \mu s$, $\bar{x} = 33.16 \mu s$
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.

```javascript
const derivedKey = await mfkdf.deriveKey(3SW.parskey(keyPolicy), {
  password: mfkdf.deriveFactors.password("Fr0b4d4@on"),
  http: mfkdf.deriveFactors.http(365287),
  recovery: mfkdf.deriveFactors.recovery("9b31b4b4-3b76-4bad-96dd-3bbdf7b346c4")
});

console.log(derivedKey.key.toString('hex')) // -> 34d2b4b438e2671c96a377f23771
```

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

https://demo.mfkdf.com

https://wallet.mfkdf.com
PBKDF2 is also used in...
MFKDF Summary

USABILITY & FACTOR COMPATIBILITY
- Knowledge
- Soft Tokens
  - USB Key
  - Out-of-Band
- Intrinsic

EXPONENTIAL SECURITY
- 14 bits
- 20 bits
- 160 bits
- 14 bits = 16s
- 194 bits ≈ 10^8 yrs

CLIENT-SIDE RECOVERY
- AND
- OR
- AND

POLICY ENFORCEMENT

HIGHLY PERFORMANT

NEW & EXISTING APPLICATIONS
Thanks!

https://mfkdf.com
https://arxiv.org/abs/2208.05586
https://github.com/multifactor/mfkdf