inovex

How to Crypto I - Korrekte Auswahl von Algorithmen/Bibliotheken/Protokollen

Bernd Kaiser

Agenda

- Passwörter Speichern / Hashen
- File Hashing
- Secret-Key Crypto (symmetrisch)
- Public-Key Crypto (asymmetrisch)
- Authentifizierte und Verschlüsselte Kommunikation

Passwörter Speichern / Hashen

Passwörter speichern ist eine Kunst:

- **niemals im Klartext**
- immer mit einer Prise Salz
- manchmal mit ein bisschen Pfeffer
- und immer mit den aktuellsten PW-Hashing Funktionen (KDF)

OWASP Password Storage Cheat Sheet

Passwörter Speichern / Hashen

1. Passwort
2. Passwort + Salt
3. Hash Function + Parameter festlegen
4. hash(Passwort + Salt)

5. Output mit Parametern encoden (ggfs. base64)

1. S3©®e7
2. S3©®e7w1KFErLD
3. Argon2id Revision 19 m=16,t=2,p=1,l=16
4. 513cf68b6c9400cabfde0a2673a87d71
5. $argon2id$v=19$m=16,t=2,p=1$dzFLRkVyTEQ$UTz2i2yUAMq/3gomc6h9cQ

inovex

👑 Argon2

- Key Derivation Function (KDF)
- Sieger der Password Hashing Competition 2015
- Individuell konfigurierbar:
  - Iterationen
  - Speicherbedarf
  - Grad der Parallelität
- Variante: **Argon2id** für den allgemeinen Gebrauch

Password Storage Cheat Sheet
BSI Technische Richtlinie Kryptographische Verfahren: Empfehlungen und Schlüssellängen

Argon2id Parameter

- Auswahl der Parameter muss stets reevaluiert werden
- Abhängig von eingesetzter Hardware
- Parameterbestimmung z.B. mit kratos oder OWASP Password Storage Cheat Sheet

Default Werte:

- Salt-Länge: 16 Bytes
- Key-**L**änge: 32 Bytes
- ***M***emory: 12288
- ***P***arallelism: 1
- I***t***erations: 3

Scrypt & PBKDF2

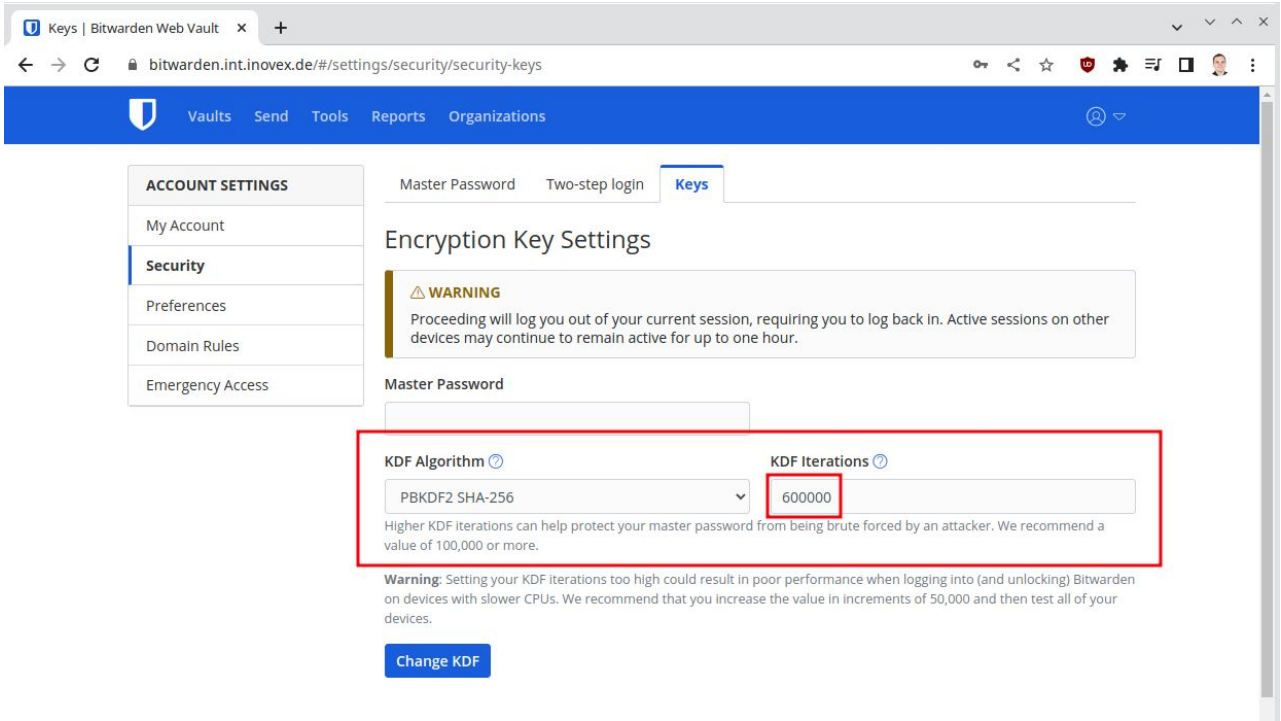
Scrypt

❌ Geringe Speicherhärte
✅ ASIC Implementierungen dank “Crypto”

PBKDF2

❌ schwach gegen GPU Angriffe
✅ FIPS zertifiziert

Exkurs: inovex Bitwarden KDF Iterations



Bitwarden design flaw: Server side iterations - Password Storage Cheat Sheet

inovex

Password Hashing Libraries

- C++
  - libsodium
- Python
  - passlib
- JavaScript (asm.js & wasm)
  - libsodium.js
- Java
  - Spring Security
- C#
  - ASP.NET Core Identity Framework

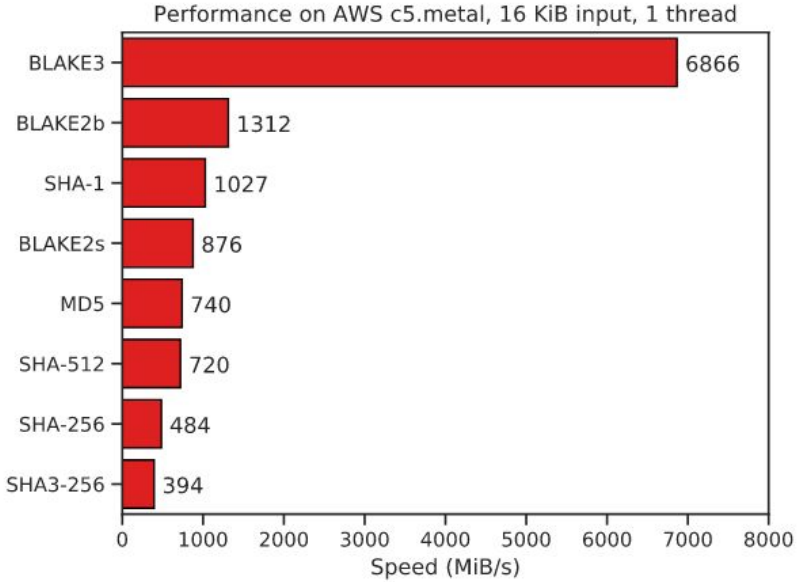
Generisches Hashing

⛔ **Nicht für Passwörter!**

- Fingerabdruck fester Länge für eine beliebig lange Nachricht
- Mögliche Einsatzzwecke
  - Prüfung der Datei-Integrität
  - Erstellung eindeutiger Bezeichner zur Indizierung beliebig langer Daten

Generisches Hashing - Algorithmen

- 👑 BLAKE3 / BLAKE2b
- 🪓 Gimli Hash
- 🐌 SHA3 FIPS
- 👴 SHA-{256,512}FIPS



⛔ **Nicht für Passwörter!**

Source: https://github.com/BLAKE3-team/BLAKE3

inovex

Generic Hashing Libraries

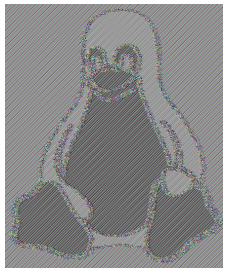
- C / C++
  - libsodium
  - libhydrogen
- Python 3.6+
  - hashlib
- Go
  - x/crypto
- Rust / C
  - BLAKE3-team/BLAKE3

⛔ **Nicht für Passwörter!**

Secret-Key Crypto (Symmetrische Verschlüsselung)

- Eine Nachricht wird mit dem selben Key und derselben Nonce ver- und entschlüsselt

- Verschiedene Modes
  - ***Authenticated*** **encryption with associated data (AEAD)**
  - Cipher Block Chaining (CBC)
  - ~~Electronic codebook (ECB)~~



ECB Tux

The Order of Encryption and Authentication for Protecting Communications

Secret-Key Crypto - Mögliche Pitfalls

Unterschiedlich Einschränkungen je nach Algorithmus

- Max Bytes pro (Key,Nonce)
- Max Bytes pro Key
- Max Anzahl von Verschlüsselungen pro Key (für verschiedene Message Sizes)
- Unterschiedliche Nonce-Anforderungen

AEAD Ciphers - XChaCha20-Poly1305

- ✅ Einfache Nonce Verwaltung (zufällige 24B)
- ✅ Keine Beschränkung der Cipher Text Länge (2^64B)
- ✅ Bis zu 2^41 Session Nachrichten selbst bei großen Nachrichten (64 GB)
- ✅ Schnell auch ohne Hardware-Unterstützung
- ✅ Bald ein IETF Standard

- ❌ Nicht FIPS zertifiziert
- ❌ Keine Hardware-Unterstützung

AEAD Ciphers - AES-GCM-{128, 256}

✅ Moderne CPUs haben meist AES Hardware-Unterstützung
✅ FIPS zertifiziert

❌ 16B Nonces zu kurz für zufällige Nonce-Verwaltung
❌ Seitenkanalattacken möglich ohne Hardware-Unterstützung
❌ Maximale Cipher Text länge 64 GB

https://soatok.blog/2020/05/13/why-aes-gcm-sucks/

AEAD Ciphers - AES-GCM-SIV

- Nonce-Misuse Resistent
- Encryption 70% Geschwindigkeit
- Decryption 100% Geschwindigkeit

Symmetrische Verschlüsselung - Libraries

- Rust
  - RustCrypto/AEADs
- Go
  - x/crypto
- Python
  - pynacl
- C/C++
  - libsodium

Public-Key Crypto (asymmetrisch Verschlüsselung)

- Signaturen
- (DH) Key Austausch
- Public-Key Verschlüsselung

Asymmetrische Signaturen

- Private Key + Message = Signatur
- Mit zugehörigen Public Key + Message kann Signatur verifiziert werden
- Anwendungsmöglichkeiten
  - Kommunikationspartner identifizieren
  - Downloads/Software verifizieren

Signing - Algorithmen

- 👑 EdDSA - ed25519
- 📜 ECDSA - NIST P-256[FIPS] (secp256r1)
- 👴 RSA[FIPS] (Seriously, stop using RSA)

Signing - ed25519

✅ 128-bit Security Level
✅ Deterministische Nonces (keine Nonce reuse möglich)
✅ “Exclusive Ownership” (1 Nachricht ➔ 1 Signature)

❌ Nicht FIPS zertifiziert
❌ Nicht geeignet für Ring-Signaturen oder Zero-Knowledge Proofs

inovex

Practical Cryptography for Developers - EdDSA and Ed25519

Signing - NIST P-256 (ECDSA, secp256r1)

✅ 128-bit Security Level
✅ FIPS zertifiziert
✅ Prime Order Group
✅ Public Key kann extrahiert werden

❌ Viele Optionen (Curve Points, Hash Function, Nonce) ➔ viele Fallstricke

PS3 Epic Fail

Signing - Libraries / Frameworks

- Minisign
- C/C++
  - libsodium
- Python
  - pynacl
- Go
  - crypto/ed25519

Diffie-Hellman Schlüsselaustausch

- X25519 (based on curve25519)
- NIST P-256 (secp256r1)

Hands-on: X25519 Key Exchange

Asymmetrische Verschlüsselung

- Message von Alice für Bob
- Alice nutzt Bob’s Public Key zum verschlüsseln
- Optional: Bob nutzt Alice’s Public Key zur Verifikation
- Bob entschlüsselt Message mit seinem Private Key

Asymmetrische Verschlüsselung - Formate

NaCl / libsodium - crypto_box:

- Schlüsselaustausch: X25519
- AEAD Verschlüsselung: XSalsa20-Poly1305

age:

- Schlüsselaustausch: X25519
- AEAD Verschlüsselung: ChaCha20-Poly1305

Asymmetrische Verschlüsselung - Formate

✅ Nacl / libsodium
✅ age
✅ libhydrogen

❌ Eigene Formate
❌ RSA

Asymmetrische Verschlüsselung - Libraries

- Python
  - pynacl
- Go
  - x/crypto/nacl/box
  - age
- C#
  - libsodium
- Rust
  - rage
  - crypto_box

Authentifizierte und Verschlüsselte Kommunikation

- Benutzt TLS v1.2 oder besser **v1.3**
- Bei TLS v1.2 sichere Cipher Suites wählen

“don't roll your own cryptographic protocol”

TLS 1.3: Every byte explained and reproduced

Resources

- Password Storage Cheat Sheet
- Practical Cryptography for Developers
- Cryptographic Right Answers
- Bitwarden design flaw: Server side iterations
- Why AES-GCM Sucks
- Seriously, stop using RSA
- PS3 Epic Fail
- Three Lessons from Threema
- TLS 1.3: Every byte explained and reproduced