Chapitre 4 : Protocoles d'Authentification
Authentication Protocols & Password Security

Cours de Sécurité Informatique - Niveau Universitaire
Partie 2 : Protocoles d'Authentification

12 janvier 2026

Table des matières

1 Introduction

1.1 Objectifs du chapitre

L'authentification est le processus de vérification de l'identité d'une entité (utilisateur, système, service). Ce chapitre couvre :

— Construire des protocoles d'authentification à partir de primitives cryptographiques
— Authentification par mot de passe : défis et vulnérabilités
— Attaques cryptanalytiques sur les protocoles à mot de passe
— Protocoles Challenge-Response
— Authentification multi-facteurs (MFA)

1.2 Définitions

Définition

Authentification : Processus de vérification qu'une entité est bien celle qu'elle prétend être.
Trois facteurs d'authentification :

— **Quelque chose que vous savez** : mot de passe, PIN
— **Quelque chose que vous avez** : token, carte à puce, smartphone
— **Quelque chose que vous êtes** : biométrie (empreinte, iris, voix)

Distinction importante :

— **Authentification** : Vérifier l'identité
— **Autorisation** : Déterminer les permissions/droits d'accès

2 Authentification par mot de passe

2.1 Modèle basique

Le modèle le plus simple : l'utilisateur prouve son identité en fournissant un mot de passe secret.

Protocole naïf :

1. Client envoie : $(username, password)$
2. Serveur vérifie : $password \stackrel{?}{=} stored\_password$

Avertissement

Problèmes majeurs :

— Transmission en clair : vulnérable à l'écoute (sniffing)
— Stockage en clair : une compromission du serveur révèle tous les mots de passe
— Pas de protection contre le rejeu (replay attack)

2.2 Stockage sécurisé : Hachage des mots de passe

Principe : Stocker $h = H(password)$ au lieu de $password$ en clair.
Vérification :

1. Client envoie : $(username, password)$

2. Serveur calcule : $h' = H(password)$
3. Serveur compare : $h' \stackrel{?}{=} h$

Propriété requise : H doit être une fonction de hachage cryptographique (SHA-256, SHA-3, etc.)

Avertissement

Attaque par dictionnaire : Si les mots de passe sont faibles, l'attaquant peut pré-calculer $H(w)$ pour tous les mots w du dictionnaire et comparer avec les hachés volés.
Rainbow Tables : Tables pré-calculées optimisées pour inverser les fonctions de hachage.

2.3 Salage (Salting)

Solution : Ajouter un sel aléatoire unique par utilisateur.
Stockage : $(salt, h)$ où $h = H(salt \| password)$
Vérification :

1. Serveur récupère $(salt, h)$ pour l'utilisateur
2. Serveur calcule : $h' = H(salt \| password)$
3. Serveur compare : $h' \stackrel{?}{=} h$

Avantages :

— Les rainbow tables deviennent inutiles (un sel différent par utilisateur)
— Deux utilisateurs avec le même mot de passe auront des hachés différents
— Force l'attaquant à attaquer chaque mot de passe individuellement

2.4 Fonctions de dérivation de clés (KDF)

Pour ralentir les attaques par force brute, utiliser des fonctions spécialement conçues :

— **PBKDF2** : Itère H de nombreuses fois (ex : 100,000 itérations)
— **bcrypt** : Fonction coûteuse en calcul, paramètre de coût ajustable
— **scrypt** : Coûteuse en mémoire (résiste aux attaques GPU/ASIC)
— **Argon2** (recommandé) : Gagnant du Password Hashing Competition (2015)

Exemple

Exemple bcrypt :
Stocker : $hash = \text{bcrypt}(password, salt, cost = 12)$
Le paramètre $cost = 12$ signifie $2^{12} = 4096$ itérations internes.

3 Protocoles Challenge-Response

3.1 Principe

Éviter de transmettre le mot de passe (même haché) sur le réseau.
Idée : Le serveur envoie un challenge aléatoire, le client prouve qu'il connaît le secret en répondant correctement.

3.2 Protocole basé sur MAC

Setup : Client et serveur partagent une clé secrète k (dérivée du mot de passe)
Protocole :

1. Client $\rightarrow$ Serveur : *username*
2. Serveur $\rightarrow$ Client : *challenge* (nonce aléatoire)
3. Client $\rightarrow$ Serveur : $response = \text{MAC}_k(challenge)$
4. Serveur vérifie : $response \stackrel{?}{=} \text{MAC}_k(challenge)$

Avantages :

— Le secret n'est jamais transmis
— Protection contre le rejeu (challenge différent à chaque fois)

3.3 Protocole SCRAM (Salted Challenge Response Authentication Mechanism)

Utilisé notamment par MongoDB, PostgreSQL.
Caractéristiques :

— Authentification mutuelle (client et serveur s'authentifient)
— Utilise PBKDF2 pour dériver les clés
— Résiste aux attaques par dictionnaire même si l'attaquant observe le trafic
— Le serveur ne stocke pas le mot de passe en clair

4 Attaques sur les protocoles à mot de passe

4.1 Attaque par dictionnaire en ligne

Méthode : L'attaquant essaie directement les mots de passe courants contre le service.
Défenses :

— **Rate limiting** : Limiter le nombre de tentatives par IP/compte
— **Account lockout** : Bloquer le compte après n échecs (attention au DoS)
— **CAPTCHA** : Après plusieurs échecs
— **Monitoring** : Détecter les patterns d'attaque

4.2 Attaque par dictionnaire hors ligne

Scénario : L'attaquant obtient la base de données des hachés (breach)
Méthode : Calculer $H(w)$ pour chaque mot w du dictionnaire et comparer
Défenses :

— Salage (obligatoire)
— KDF lentes (PBKDF2, bcrypt, scrypt, Argon2)
— Politique de mots de passe forts

4.3 Credential Stuffing

Méthode : Utiliser des identifiants volés d'un site sur d'autres sites
Problème : Réutilisation de mots de passe entre services
Défenses :

— Éducation des utilisateurs (ne pas réutiliser les mots de passe)
— Gestionnaires de mots de passe
— Détection de connexions anormales (géolocalisation, device fingerprinting)

4.4 Phishing

Méthode : Tromper l'utilisateur pour qu'il révèle son mot de passe sur un faux site
Défenses :

— Formation des utilisateurs
— Authentification multi-facteurs (MFA)
— Certificats TLS et validation du domaine

4.5 Man-in-the-Middle (MitM)

Méthode : Intercepter la communication entre client et serveur
Défenses :

— TLS/HTTPS obligatoire (chiffrement de bout en bout)
— Certificate pinning
— Protocoles Challenge-Response (pas de transmission du secret)

5 Authentification multi-facteurs (MFA)

5.1 Principe

Combiner plusieurs facteurs d'authentification indépendants.
Exemple 2FA : Mot de passe (ce que vous savez) + Code SMS (ce que vous avez)

5.2 Types de seconds facteurs

1. **SMS/OTP** : Code à usage unique envoyé par SMS
   — **Avantage** : Simple, compatible avec tous les téléphones
   — **Inconvénient** : Vulnérable au SIM swapping, interception SMS
2. **TOTP (Time-based OTP)** : Google Authenticator, Authy
   — Code généré localement basé sur l'heure et une clé secrète
   — Standard : RFC 6238
   — Plus sécurisé que SMS
3. **Hardware tokens** : YubiKey, clés de sécurité FIDO2
   — **Avantage** : Très sécurisé, résiste au phishing
   — **Inconvénient** : Coût, peut être perdu
4. **Biométrie** : Empreinte digitale, reconnaissance faciale
   — **Avantage** : Pratique, difficile à voler
   — **Inconvénient** : Impossible de "changer" si compromis
5. **Push notifications** : Duo, Microsoft Authenticator
   — Notification sur smartphone à approuver
   — Vulnérable à la fatigue d'authentification (MFA fatigue)

5.3 FIDO2 et WebAuthn

Standard moderne : Authentification sans mot de passe (passwordless)
Principe :

— Utilise la cryptographie à clé publique
— Clé privée stockée dans un token hardware (YubiKey) ou TPM
— Résiste au phishing (validation du domaine intégrée)
— Pas de secret partagé avec le serveur

6 Protocoles avancés

6.1 Password-Authenticated Key Exchange (PAKE)

Objectif : Établir une clé de session à partir d'un mot de passe partagé
Protocoles :

— **SRP (Secure Remote Password)** : Utilisé par Apple iCloud

— **SPAKE2** : Standardisé, simple et efficace

— **OPAQUE** : État de l'art, en cours de standardisation IETF

Propriétés :

— Résiste aux attaques par dictionnaire en ligne

— Le serveur ne stocke pas de vérificateur permettant une attaque hors ligne

— Forward secrecy

6.2 Zero-Knowledge Password Proof (ZKPP)

Principe : Prouver la connaissance du mot de passe sans le révéler
Application : 1Password, Bitwarden (architecture zero-knowledge)

7 Travaux pratiques

7.1 Exercices théoriques

1. **Analyse de protocole** : Montrer que le protocole naïf (envoi du mot de passe en clair) est vulnérable au rejeu.
2. **Calcul de complexité** : Si un attaquant peut tester 10^9 mots de passe/seconde, combien de temps faut-il pour casser un mot de passe de 8 caractères alphanumériques ?
3. **Salage** : Expliquer pourquoi le salage rend les rainbow tables inefficaces.
4. **TOTP** : Décrire le fonctionnement de TOTP. Quel est le rôle du temps dans ce protocole ?
5. **MFA bypass** : Quelles sont les attaques possibles contre MFA par SMS ?

7.2 Exercices pratiques

Les notebooks suivants implémentent et explorent les concepts de ce chapitre :

— `04_demo_password_hashing.ipynb` : Hachage avec sel, PBKDF2, bcrypt

— `04_demo_totp.ipynb` : Implémentation de TOTP (RFC 6238)

— `04_demo_challenge_response.ipynb` : Protocole challenge-response avec HMAC

— `04_exercices.ipynb` : Exercices guidés sur l'authentification

Exercices proposés :

1. Implémenter le stockage sécurisé de mots de passe avec bcrypt
2. Créer un générateur TOTP compatible avec Google Authenticator
3. Simuler une attaque par dictionnaire et mesurer l'impact du coût bcrypt
4. Analyser des dumps de bases de données réelles (rockyou.txt) et identifier les patterns de mots de passe faibles

8 Ressources complémentaires

8.1 Standards et RFCs

— RFC 6238 : TOTP (Time-Based One-Time Password)

— RFC 2898 : PBKDF2 (Password-Based Key Derivation Function)

— FIDO Alliance : Standards FIDO2/WebAuthn

— NIST SP 800-63B : Digital Identity Guidelines (Authentication)

8.2 Outils pratiques

— **John the Ripper** : Outil de cracking de mots de passe

— **Hashcat** : Cracking GPU-accéléré

— **Have I Been Pwned** : Base de données de breaches (haveibeenpwned.com)

— **zxcvbn** : Librairie d'estimation de force de mots de passe (Dropbox)

8.3 Lectures recommandées

— Bonneau et al. (2012). "The Quest to Replace Passwords"

— Herley & van Oorschot (2012). "A Research Agenda Acknowledging the Persistence of Passwords"

— OWASP Authentication Cheat Sheet

9 Conclusion

Points clés :

— Ne jamais stocker ou transmettre des mots de passe en clair

— Toujours utiliser un sel unique par utilisateur

— Utiliser des KDF lentes (Argon2, bcrypt, scrypt)

— L'authentification multi-facteurs est essentielle pour la sécurité moderne

— Les protocoles Challenge-Response évitent la transmission du secret

— Éduquer les utilisateurs sur les bonnes pratiques (gestionnaires de mots de passe, pas de réutilisation)

Évolution : Vers une authentification sans mot de passe (passwordless) avec FIDO2/WebAuthn.

"The only secure password is the one you can't remember."
— Troy Hunt, Have I Been Pwned