

ZERO KNOWLEDGE ARCHITECTURE

La Webapp sécurisée est-elle possible ?

► <http://talks.m4dz.net/zero-knowledge-architecture/?format=short>

PARTAGER : QUOI, COMMENT, ET AVEC QUI ?



Cloud everywhere

Qui stocke quelles informations ?

L'information transite (beaucoup)

Reprendre le contrôle



QUI PEUT ACCÉDER AUX DONNÉES



QUELLES INFORMATIONS SONT PARTAGÉES



QUELLES SONT LEURS DURÉES DE VIE

« **ZKA** est un design d'application qui permet de fournir des accès à la donnée personnelle pour les applications clientes en garantissant que ces services n'auront jamais accès à des contenus en clair, sans permission.

Concepts

- Authentication zero-proof
- End-to-end encryption
- Encrypted Data Only
- No-naive approach

Comment ça fonctionne ?



PARTAGER : QUOI, COMMENT, ET AVEC QUI ?

Zero-knowledge

Zero-knowledge

1.  création d'un mot de passe

Zero-knowledge

1.  création d'un mot de passe
2.  création d'une clef intermédiaire, chiffrée par ce mot de passe, basé sur le CER racine de l'app sur le serveur

Zero-knowledge

1.  création d'un mot de passe
2.  création d'une clef intermédiaire, chiffrée par ce mot de passe, basé sur le CER racine de l'app sur le serveur
3.  création de deux paires de clefs RSA : une pour l'auth (signature), l'autre pour la data (chiffrement), chiffrées avec la clef intermédiaire

Zero-knowledge

1.  création d'un mot de passe
2.  création d'une clef intermédiaire, chiffrée par ce mot de passe, basé sur le CER racine de l'app sur le serveur
3.  création de deux paires de clefs RSA : une pour l'auth (signature), l'autre pour la data (chiffrement), chiffrées avec la clef intermédiaire
4.  les clefs publiques sont envoyées sur le serveur + hash des clefs privées uniquement

Zero-knowledge

1.  création d'un mot de passe
2.  création d'une clef intermédiaire, chiffrée par ce mot de passe, basé sur le CER racine de l'app sur le serveur
3.  création de deux paires de clefs RSA : une pour l'auth (signature), l'autre pour la data (chiffrement), chiffrées avec la clef intermédiaire
4.  les clefs publiques sont envoyées sur le serveur + hash des clefs privées uniquement
5.  les clefs privées sont stockées côté client, déchiffrées avec la clef intermédiaire, elle-même déchiffrée avec le mdp utilisateur

AUTH (ZKP)

Clefs

- un certificat intermédiaire par service
- deux paires de clefs par service
- la paire de signature sert à l'authentification

Preuve à connaissance nulle

- le service qui souhaite s'authentifier se présente au serveur
- le serveur teste sa clef privée de signature via un handshake
- le serveur garantit au client la bonne auth du service

Sécurité

- il n'y a pas d'échange de mot de passe
- les clefs sont révocables via les CER intermédiaires en cas de compromission

E2EE

Chiffrement

- dans le client uniquement
- en utilisant la clef publique du service qui demande l'accès
- en faisant du *key wrapping* de clef synchrone (IDEA, par exemple)

Déchiffrement

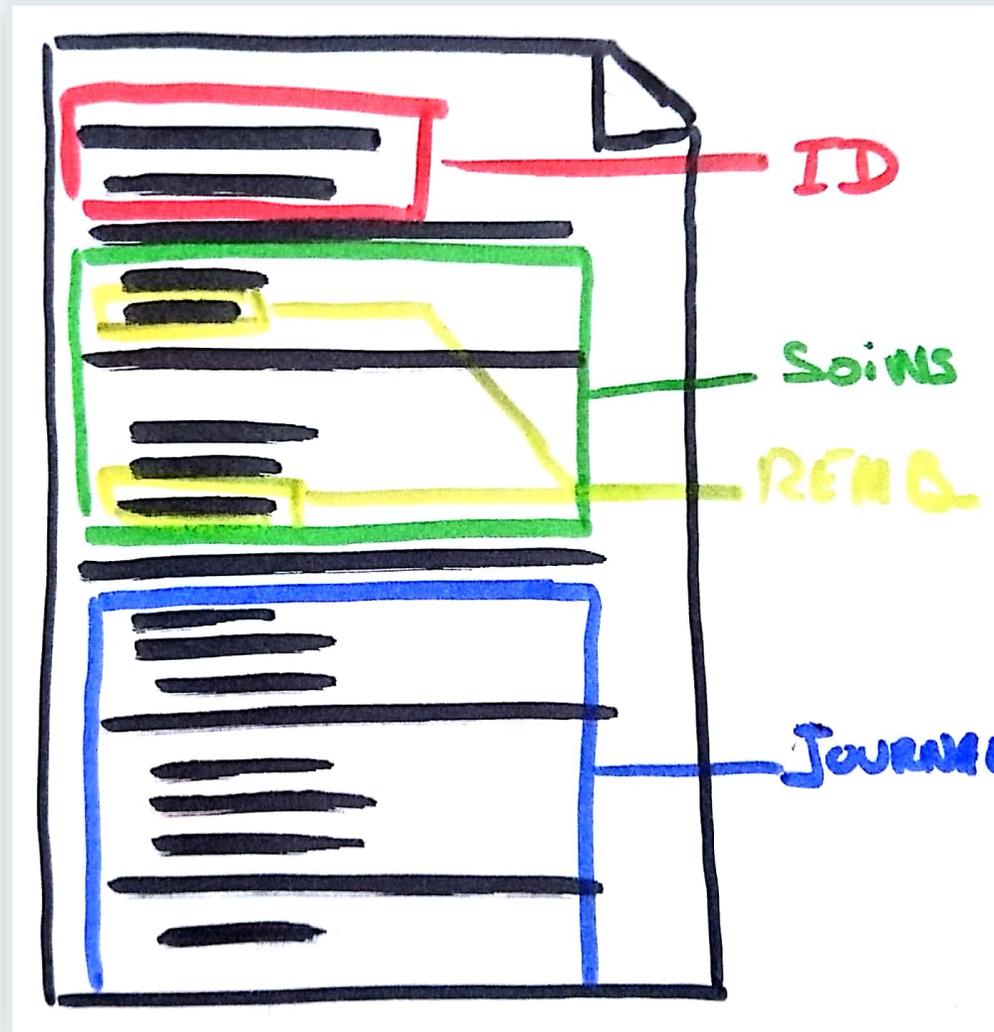
- dans le service
- en utilisant la clef privée pour récupérer la clef synchrone

Sécurité

- chaque clef synchrone est à usage unique data / service
- la clef synchrone porte l'expiration (datetime token)

APPROCHE NON-NAÏVE

Document tree Data Blob



Sécurité

- pas de partage global des documents
- chaque blob est chiffré isolément, par service, avec une clef unique
- la granularité est la plus fine possible
- le stockage se fait dans le cloud, chiffré, sans connaissance des clefs
- les accès aux ressources interdites sont impossibles



COMMENT C'EST POSSIBLE ?

ÉTAT DE L'ART

COMMENT C'EST POSSIBLE ?

C'est compliqué



ZKA s'appuie sur la *confiance* dans
l'app cliente

Implémentations

- SignalProtocolKit::SessionBuilder
- CossackLabs (PoC)

Mobile / Desktop

- code compilé
- stockage de clefs sécurisé
- environnement contrôlé
- intrusions prévenues

Coté Web ?

Sécuriser la couche crypto du navigateur

CORS

- protège des requêtes vers des domaines non-reconnus
- prévient des injections depuis des ressources extérieures
- interdit l'écriture sauvage dans le document

CSP

- autorise explicitement les ressources (JS, assets, etc)
- prévient l'injection XSS / data
- protège l'intégrité de l'app

SRI

- vérifie la signature des assets
- protège du MITM
- garantit l'intégrité des ressources exécutées

Referrer-Policy

- évite la fuite des URI privées
- isole les URLs de l'app
- protège du tracking malicieux

Key-storage

- basé sur WebCrypto
- avec File-API
- en proposant une mécanique d'export des clefs chiffrées / CER intermédiaires

WebAssembly

Prévient la lecture du code exécuté et rend l'extraction de données complexe

ZKA (POUR LE WEB, MAIS PAS SEULEMENT)

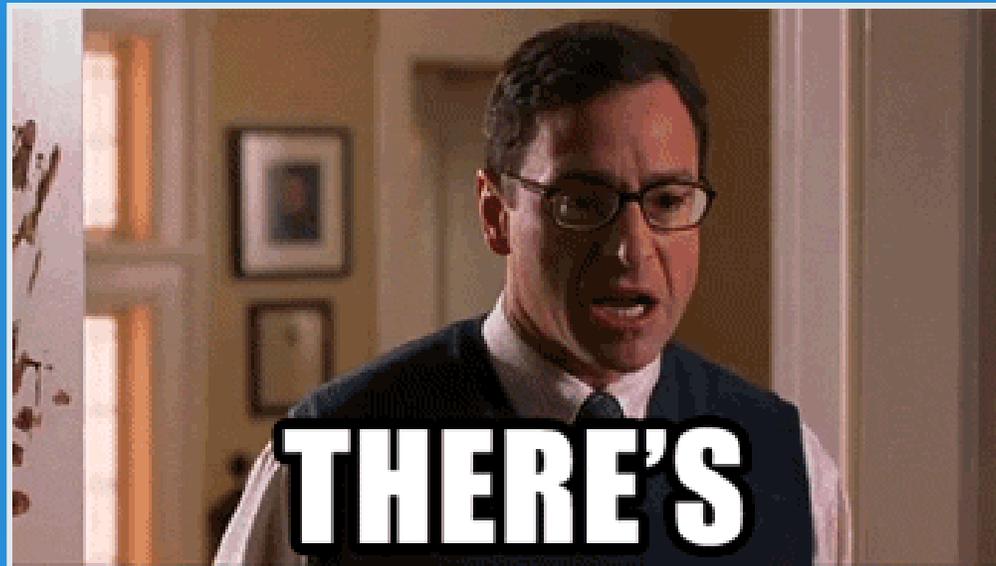


Minimiser les risques

- fuite de données → Chiffrement
- escalades de privilèges → Chiffrement
- usurpation d'ID → ZKP (Chiffrement)
- limitation de la zone d'attaque

Dans quels contextes ?

- logs
- complex docs
- filesystems
- remote storage



À qui choisissons-nous de faire confiance ?

- Aux constructeurs ?

À qui choisissons-nous de faire confiance ?

- Aux constructeurs ?
- Aux implémenteurs ?

À qui choisissons-nous de faire confiance ?

- Aux constructeurs ?
- Aux implémenteurs ?
- Aux éditeurs ?

À qui choisissons-nous de faire confiance ?

- Aux constructeurs ?
- Aux implémenteurs ?
- Aux éditeurs ?
- Aux utilisateurs ?

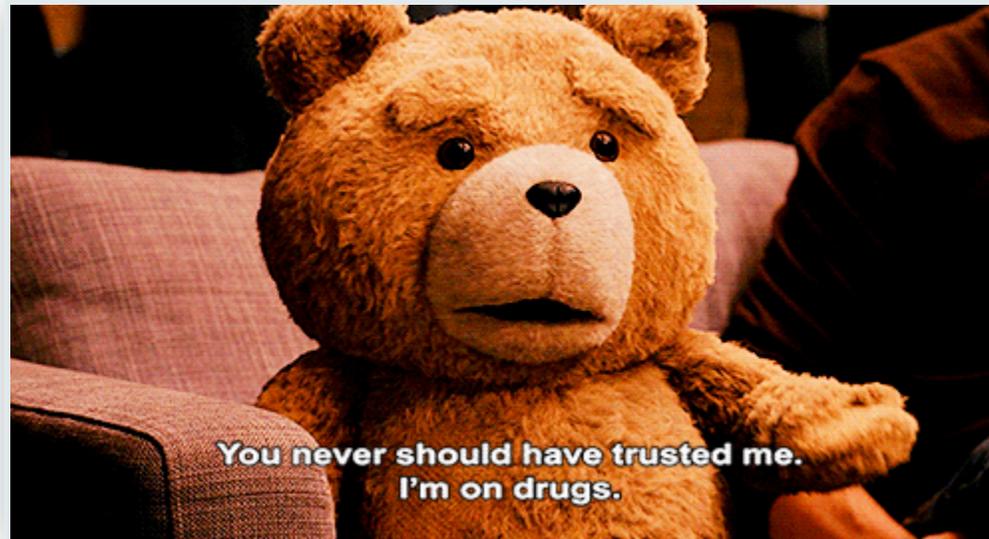


NOUS AVONS BESOIN D'AUDITABILITÉ

Open source, audits publics, structures publiques, rapports autonomes...

ZKA

- complexe et coûteux
- le Web en est capable
- ce n'est qu'un moyen
- trou de confiance





M4DZ

Paranoïd Web Dino & Tech Evangelist

m4dz.net | [@m4d_z](https://twitter.com/m4d_z) | PGP [0xD4627C417D969710](https://pgp.mit.edu/pks/lookup?op=index&search=0xD4627C417D969710)



www.alwaysdata.com



QUESTIONS ?



> <http://talks.m4dz.net/zero-knowledge-architecture/?format=short>
disponible sous licence  CC BY-SA 4.0