SNT Chat

PRIVATE MESSAGING

DOCUMENT PUBLIC · TRUST CENTER

Security Whitepaper

Architecture, cryptographie, modèle de menace et transparence pour la messagerie SNT Chat · beta Android et feuille de route E2EE.

VERSION

1.0.0

DATE

6 août 2026

CLASSIFICATION

Public · sntchat.com

Résumé exécutif

SNT Chat est une messagerie privée en beta (Android V1) centrée sur l'identité **`@username`**, la transparence (**Trust Center** public) et une feuille de route **chiffrement bout en bout** alignée sur les standards reconnus (Signal Protocol, AES-256-GCM, X25519).

Ce whitepaper décrit l'**implémentation visée et l'état honnête du produit** au moment de la version **1.0.0** du document : ce n'est pas une promesse marketing floue. Les éléments non livrés (E2EE complet sur tous les flux, iOS App Store, audit externe) sont listés explicitement.

Pilier	Beta V1 / roadmap
Messagerie	Chats, demandes d'ami, `@username`, Security Center
Chiffrement messages	Double Ratchet (Signal), déploiement progressif
Pièces jointes	Chiffrement côté client avant envoi stockage
Transport	TLS 1.3 (HTTPS)
Identité	Firebase Auth + profil SNT ID
Auth renforcée	2FA TOTP, passkeys (Android), OAuth
Vérification contact	Numéros de sécurité, scan QR
Transparence	Trust Center, schémas, statut services
RGPD	Export, suppression compte, politique publique
App Check	Attestation app (Play Integrity, etc.)

1. Introduction et périmètre

1.1 Objectif du service

SNT Chat permet à un utilisateur de :

- Discuter en privé avec des contacts identifiés par **`@username`** ;
- Contrôler son profil public et ses demandes d'ami ;
- Vérifier l'identité d'un interlocuteur (numéro de sécurité, QR) ;
- Consulter la documentation sécurité publique et l'état des services ;
- Recevoir des alertes push **sans exposer le corps des messages** sur le réseau FCM.

SNT Chat **n'est pas** une application de stockage cloud grand public type « sauvegarde photos illimitée » : le cloud sert aux **données de compte, métadonnées de chat et pièces jointes chiffrées**, pas à

remplacer un drive généraliste.

1.2 Périmètre document v1.0.0

Inclus dans la beta Android visée :

- Application Flutter, flavor production ;
- Inscription personnelle : `@username`, e-mail, mot de passe, PIN appareil ;
- Auth e-mail, Google, Facebook, GitHub (selon plateforme) ;
- Passkeys WebAuthn sur Android ;
- 2FA TOTP + codes de récupération ;
- Trust Center (`sntchat.com/trust`) et whitepaper public ;
- Notifications push génériques (pas de contenu message dans le payload FCM) ;
- Réservation `@username` via collection Firestore dédiée.

Exclus ou en cours (annoncés publiquement) :

- Chiffrement bout en bout sur **100 %** des messages en production ;
- Application iOS / Apple Sign-In store ;
- Audit de sécurité indépendant (objectif post-beta) ;
- Certifications ISO 27001 / SOC 2 ;
- Bug bounty rémunéré (préparation).

1.3 Principes de conception

1. **Zero-knowledge (objectif)** : le backend ne doit pas pouvoir lire le contenu des messages ni des fichiers chiffrés côté client.
2. **Défense en profondeur** : TLS, règles Firestore strictes, App Check, durées limitées sur URLs signées.
3. **Transparence** : ce document, pages Trust, changelog, statut automatisé.

4. **Minimisation** : collecte limitée aux métadonnées nécessaires au service.

2. Modèle de menace

2.1 Adversaires considérés

Adversaire	Capacité	Mitigation
Opérateur malveillant	Accès Firestore, stockage objet, logs	Chiffrement client, pas de clés privées sur serveur
Attaquant réseau (MITM)	Interception HTTPS	TLS 1.3, stacks OS
Compte Firebase compromis	Lecture règles Firestore	Isolation par UID, pas d'accès cross-user
Phishing identifiants	Vol e-mail / MDP	Passkeys, 2FA TOTP
Malware sur appareil	Lecture mémoire / clés	Verrou PIN/biométrie, limite du modèle E2EE
Usurpation de contact	Mauvaise personne	Numéros de sécurité, QR, Security Center
Client non autorisé (script)	Appels API sans app	App Check (Play Integrity, etc.)

2.2 Hors périmètre explicite

- Récupération si perte simultanée des facteurs de récupération et du verrou appareil ;
- Appareil rooté / OS compromis au démarrage ;

- Résistance quantique complète (roadmap hybride X25519 + ML-KEM).

3. Architecture système

3.1 Vue d'ensemble

```
[Appareil Android / iOS]
    | UI Flutter · moteurs crypto / chat locaux
    | Chiffrement messages & PJ (progressif E2EE)
    ▼
[Firebase Auth]     identité, OAuth, passkeys
[Cloud Firestore]   conversations, messages, profils (UID-scoped)
[Cloud Functions]   logique métier, invites, RGPD, FCM
[Backblaze B2]      blobs chiffrés (pièces jointes)
[FCM]               push génériques (sans corps de message)
[Firebase Hosting]  site · Trust Center · docs · .well-known
```

3.2 Composants

Composant	Rôle	Contenu message en clair ?
App Flutter	Chiffrement, UI, sync	Oui (sur appareil déverrouillé)
Firebase Auth	Sessions, OAuth	Non
Firestore	Métadonnées, enveloppes	Non (selon phase E2EE)
Cloud Functions	Orchestration, URLs signées, push	Non
B2	Stockage objet chiffré	Non
FCM	Alertes	Non (titre générique)

3.3 Régions

- Cloud Functions : **europe-west1** lorsque applicable ;
- Firestore : région Europe selon configuration projet ;

- Stockage objet : région EU ou US selon politique compte / conformité.

4. Cryptographie des messages

4.1 Standards retenus

Alignement **SNT Crypto Engine** (pas d'algorithmes « marketing » non standard) :

- **AES-256-GCM** : chiffrement symétrique authentifié ;
- **ChaCha20-Poly1305** : prévu selon contexte ;
- **X25519, Ed25519, HKDF** : échange et signatures ;
- **Double Ratchet (Signal)** : sessions messagerie.

Roadmap post-quantique documentée : hybride **X25519 + ML-KEM-768**, signatures **ML-DSA / SLH-DSA**.

4.2 État beta (transparence)

Le déploiement E2EE est **progressif** : certaines phases peuvent transporter des enveloppes ou métadonnées en clair côté serveur le temps de la migration. Le Trust Center (`/trust/visibility`) décrit **ce que SNT Chat voit aujourd'hui** par rapport au contenu des messages.

4.3 Clés et sessions

- Génération d'identité et **prekeys** côté client ;
- Clés privées : **flutter_secure_storage**, Android Keystore / StrongBox, Secure Enclave (iOS) ;
- **Numéro de sécurité** dérivé des clés publiques pour vérification manuelle ou QR.

4.4 Ce que le serveur ne reçoit jamais

- Clé privée long terme en clair ;
- Corps de message en clair (objectif E2EE) ;
- PIN appareil en clair ;
- Secrets TOTP en clair (stockage haché côté serveur).

5. Pièces jointes et stockage

Les fichiers joints suivent un modèle **chiffrement client puis envoi** :

1. Lecture locale du fichier ;
2. Chiffrement **AES-256-GCM** (clé dérivée / envelope selon implémentation) ;
3. Upload direct vers stockage objet via URL signée à durée limitée ;
4. Métadonnées (nom, taille, hash, clé de référence) en Firestore via Functions.

Le serveur ne reçoit **jamais** : mot de passe vault, clé maître, contenu fichier en clair.

6. Gestion des clés et verrou appareil

6.1 Identité cryptographique

- Paire de clés générée localement (CSPRNG) ;
- Clés publiques et prekeys publiées pour établir les sessions ;
- Clés privées jamais envoyées au serveur en clair.

6.2 PIN SNT (6 chiffres)

- Hash **PBKDF2** (ou équivalent) + sel aléatoire ;
- Stockage : `flutter_secure_storage` (Keychain/Keystore) ;
- Jamais synchronisé cloud ;
- Auto-verrouillage après inactivité (politique app).

6.3 Biométrie

- Déverrouillage optionnel via `local_auth` ;
- Ne remplace pas la récupération en cas de perte de facteurs.

7. Authentification et session

7.1 Méthodes V1

Méthode	Statut V1
E-mail / mot de passe	Oui
Google Sign-In	Oui
Facebook	Oui (Android)
GitHub	Oui (Android)
Apple Sign-In	Selon plateforme / store
Passkeys	Oui Android
2FA TOTP	Oui

7.2 Session Firebase

- Token JWT Firebase géré par SDK ;
- Callables exigent authentification (UID) ;
- Déconnexion : purge session + verrou app local.

7.3 Vérification compte

- Parcours profil et e-mail vérifié selon règles produit ;
- Quotas et fonctionnalités beta liés au statut compte documenté in-app.

8. Passkeys (WebAuthn / FIDO2)

8.1 Implémentation V1 (Android)

- Enrôlement et authentification via WebAuthn ;
- Clé privée dans Secure Enclave / TPM ;
- Seule la **clé publique** est enregistrée côté serveur ;
- **Origin binding** : protection anti-phishing.

8.2 App Links / Digital Asset Links

- `public/.well-known/assetlinks.json` sur Hosting ;
- Associe le domaine **sntchat.com** au package Android production.

8.3 Limites

- iOS / passkeys Safari : selon calendrier store ;
- Nécessite appareil compatible FIDO2.

9. Authentification à deux facteurs (TOTP)

9.1 Standard

- **RFC 6238** (TOTP) ;
- Codes 6 chiffres, fenêtre 30 secondes ;
- Compatible Google Authenticator, Authy, etc.

9.2 Codes de récupération

- Générés à l'activation 2FA ;
- Usage unique ;
- Stockés **hachés** (jamais en clair) ;

- Collections sensibles : **inaccessibles client** (Firestore rules).

9.3 Flux login

1. Auth primaire (e-mail, Google, passkey, etc.) ;
2. Si 2FA activé : demande code TOTP ou récupération ;
3. Vérification côté Function avant session complète.

10. Messagerie, sync et Firestore

10.1 Modèle de données

- Conversations scoped par membres authentifiés ;
- Messages stockés selon phase E2EE (enveloppes chiffrées cible) ;
- Présence et métadonnées minimisées.

10.2 Temps réel

- Listeners Firestore pour inbox et threads ;
- Pas de SDK chat tiers au cœur produit (moteur SNT).

10.3 Pièces jointes

- Référence vers blob B2 chiffré + métadonnées Firestore ;
- Téléchargement via URL signée, déchiffrement local.

11. Invitations et liens

11.1 Parrainage / invite

- Pages `/invite` et parcours documentés ;
- Pas d'exposition de contenu message dans les liens.

11.2 Deep links

- Schémas app pour ouvrir conversations après auth ;

- Validation App Links côté Android.

12. Infrastructure cloud

12.1 Firebase (Google Cloud)

- **Authentication** : identité uniquement ;
- **Firestore** : métadonnées, profils, chats ;
- **Hosting** : site vitrine, Trust Center, `.well-known` ;
- **Cloud Functions** : Node.js, région `europe-west1`.

12.2 Backblaze B2

- API S3-compatible ;
- Buckets EU / US selon conformité ;
- Credentials par région (secrets Firebase).

12.3 CDN / DNS

- Domaine **sntchat.com** ;
- Cache éventuel sur blobs **déjà chiffrés** (pas de déchiffrement CDN).

13. Règles d'accès Firestore

13.1 Principes

- Lecture / écriture client limitée au `uid` authentifié ;
- Collections sensibles (secrets 2FA, etc.) : **écriture Functions uniquement** ;
- Pas d'énumération cross-tenant ;
- Réservation `@username` : transaction atomique à l'inscription.

13.2 Exemples

Collection	Client read	Client write
`users/{uid}`	Propre UID	Champs limités
`usernames/{name}`	Contrôlé	Transaction inscription
Secrets 2FA	Non	Non

14. URLs signées Backblaze B2

14.1 Upload

- Callable génère URL PUT signée (durée courte) ;
- Client envoie directement le blob chiffré.

14.2 Download

- Callable génère URL GET signée ;
- Expiration en minutes ;
- Pas de listing bucket côté client.

15. CDN et transport

15.1 TLS

- HTTPS obligatoire (Hosting, Functions, B2) ;
- TLS 1.3 négocié par les stacks modernes.

15.2 Intégrité

- Tag GCM par fichier ;
- Hash pré-chiffrement pour intégrité debug.

16. Notifications (FCM)

16.1 Usage messagerie

- Push « nouveau message » **sans corps** dans le payload ;
- Function `onChatMessageCreated` (ou équivalent) : titre générique ;
- Token FCM lié à `uid`, révoqué à la déconnexion.

16.2 Confidentialité

- Google/Apple voient un payload minimal ;

- Aperçu détaillé lock screen : contrôle client après fetch local.

17. Firebase App Check

17.1 Objectif

- Vérifier que Firestore / Auth / Functions sont appelés depuis **l'app légitime** ;
- Réduit scripts tiers et clients falsifiés.

17.2 Providers

Plateforme	Production
Android	Play Integrity
iOS	App Attest / Device Check
Web	reCAPTCHA v3 (si applicable)

Déploiement progressif : mode Monitoring puis Enforced (voir documentation projet App Check).

18. RGPD : export et suppression

18.1 Export

- Manifeste JSON profil + inventaire métadonnées (pas de contenu déchiffré serveur) ;
- Export complet déchiffré : **côté client** lorsque applicable.

18.2 Suppression

- Parcours `/account-deletion` et in-app ;
- Séquence : stockage objet, Firestore, Auth ;
- Irréversible si pas de backup local.

18.3 Sous-traitants

- Google (Firebase), Backblaze, hébergeur DNS/CDN ;

- Politique : `sntchat.com/privacy`.

19. Journalisation et données visibles

19.1 Ce que SNT Chat peut voir

Donnée	Visible serveur ?
E-mail, UID, `@username`, plan	Oui
Horodatages, IDs conversation	Oui
Corps message (cible E2EE)	Non
Pièce jointe déchiffrée	Non
PIN / clés privées	Non

19.2 Logs

- Erreurs techniques, corrélation ;
- Pas de log intentionnel de contenu message ;
- Analytics / Crashlytics : sans contenu message.

20. Trust Center et transparence

20.1 Pages publiques

- Hub `/trust` : cryptographie, architecture, zero-knowledge, status ;
- Schémas `/security` ;
- Changelog `/changelog`.

20.2 Statut automatisé

- Endpoint `/api/status` ;
- Incidents : `/api/incidents.json` ;
- Page `/trust/status`.

20.3 Communauté

- Chaîne **@SNTCHATPROTECT**, groupe **@SNTCHATGROUP** (Telegram).

21. Divulgation responsable

21.1 Contact

- **support@sntchat.com** (objet : Security disclosure) ;
- `/.well-known/security.txt` ;
- Page `/trust/disclosure`.

21.2 Scope

- App Android beta/production, site, Functions, règles Firestore, flux chiffrement messagerie.

22. Limites connues V1

1. E2EE non universalisé sur tous les flux ;
2. Pas de pentest externe publié ;
3. iOS : parité selon calendrier store ;
4. Pas de SLA contractuel public complet ;
5. Malware local : hors garantie zero-knowledge standard.

23. Roadmap sécurité

Item	Cible
Double Ratchet généralisé	Post-beta
App Check enforced	Après monitoring
Hybride post-quantique	Roadmap moteur crypto
Pentest indépendant	Post-beta
Bug bounty rémunéré	Préparation

24. Glossaire

Terme	Définition
E2EE	Chiffrement bout en bout : seuls les participants lisent le contenu
Double Ratchet	Protocole Signal pour forward secrecy
Prekey	Clé publique prépubliée pour établir une session
Zero-knowledge	Le serveur ne possède pas les secrets de déchiffrement
TOTP	Mot de passe à usage unique basé sur le temps
Passkey	Credential WebAuthn liée au domaine / app
App Check	Attestation que la requête vient de l'app officielle

25. Références

- NIST SP 800-38D (GCM)
- RFC 6238 (TOTP)
- WebAuthn (W3C)
- RGPD (UE) 2016/679
- Documentation publique : https://sntchat.com/trust
- Architecture interne : dépôt `docs/ARCHITECTURE.md`, `docs/CRYPTO_ENGINE.md`

26. Contrôle documentaire

Champ	Valeur
Titre	SNT Chat Security Whitepaper
Version	1.0.0
Date	6 août 2026
Classification	Public
Domaine	sntchat.com
Langue	Français (EN : pages Trust)

Annexe A · FAQ technique sécurité

Q : SNT Chat peut-il lire mes messages ?
R : Consultez `/trust/visibility`. L'objectif produit est le zero-knowledge sur le contenu ; l'état exact de

déploiement E2EE est documenté publiquement.

Q : Que contient une notification push ?
R : Un libellé générique, pas le corps du message dans le payload FCM.

Q : Comment vérifier un contact ?
R : Numéro de sécurité ou scan QR dans le Security Center.

Q : Où signaler une vulnérabilité ?
R : `/trust/disclosure` et `security.txt`.

Annexe B · Scénarios de menace (synthèse)

B.1 Compte d'un autre utilisateur compromis

Impact : métadonnées, pas de déchiffrement sans clés victime.
Mitigations : 2FA, passkeys, isolation Firestore.

B.2 Fuite stockage objet

Impact : blobs chiffrés uniquement.
Mitigations : URLs signées courtes, chiffrement client.

B.3 Ingénierie sociale support

Procédure : pas de backdoor, pas de récupération des clés privées ; suppression compte si demandé (RGPD).

Signature éditoriale

Document approuvé pour publication publique par **SNT Chat · Security & Privacy**.

Contact : **support@sntchat.com** · Divulgation responsable : **sntchat.com/trust/disclosure**

SNT Chat Security & Privacy