Projet PI^2 60

Objectifs : Reproduction en plusieurs exemplaires d'un dispositif open hardware et amélioration de sa sécurité.

Dispositif open hardware choisi : TKey (support amovible d'authentification)

Equipe : 4303 : ANDRIANIRINA Fanantenana Michael, HERBRON Nathan, EL BAHI Sofien, KARA Eliza, BAOUCHI Aboubakar, ARTISI Paul

Encadrant : BECQ Aline

Date de lancement : 06/11/2025

Etat de l'art :

1. Définition du domaine et contexte

Les dispositifs d'authentification matériels occupent aujourd'hui une place centrale dans les mécanismes de protection des identités numériques et des accès sensibles. Contrairement aux solutions purement logicielles, ils isolent les opérations cryptographiques au sein d'un support externe, ce qui limite fortement les risques de compromission liés aux malwares, au phishing ou aux attaques par extraction de secrets directement sur la machine de l'utilisateur.

Un dispositif d'authentification matériel est généralement une clé physique, le plus souvent connectée en USB, NFC ou BLE, qui embarque une capacité de génération, de stockage et de manipulation de secrets cryptographiques. Ses usages typiques incluent l'authentification forte à deux facteurs (2FA), la conformité aux standards FIDO2/WebAuthn, la signature de données, la génération de clés, ou encore le scellage de secrets utilisés dans des environnements professionnels ou administratifs.

Dans ce domaine, l'open hardware occupe une position spécifique et particulièrement intéressante. Contrairement aux dispositifs propriétaires, dont le code source, la conception électronique et les mécanismes cryptographiques ne sont pas accessibles, un matériel open hardware offre une transparence complète : chacun peut en étudier le fonctionnement, vérifier l'absence de comportements indésirables, reproduire le matériel, adapter le firmware et valider les choix architecturaux. Les concepts d'auditabilité, de reproductibilité et de pédagogie sont donc au cœur de cette approche. Cela permet à la fois de renforcer la confiance dans l'artefact matériel et de faciliter son utilisation dans des contextes éducatifs ou de recherche, où la compréhension des mécanismes internes est un objectif essentiel.

La Tillitis Key1 s'inscrit pleinement dans cet écosystème. Elle se distingue des dispositifs propriétaires comme la Yubikey par son modèle entièrement ouvert, tant au niveau électronique que logiciel. Contrairement à d'autres projets open hardware tels que SoloKeys, elle repose sur une architecture minimaliste centrée sur un FPGA basse consommation et un firmware écrit en Rust, conçu pour limiter au maximum la surface d'attaque. Cette architecture en fait un support particulièrement pertinent pour un projet de reproduction matérielle et d'analyse de sécurité : elle est suffisamment complexe pour aborder des problématiques réelles d'implémentation, tout en restant suffisamment épurée et ouverte pour permettre une étude complète de son fonctionnement interne.

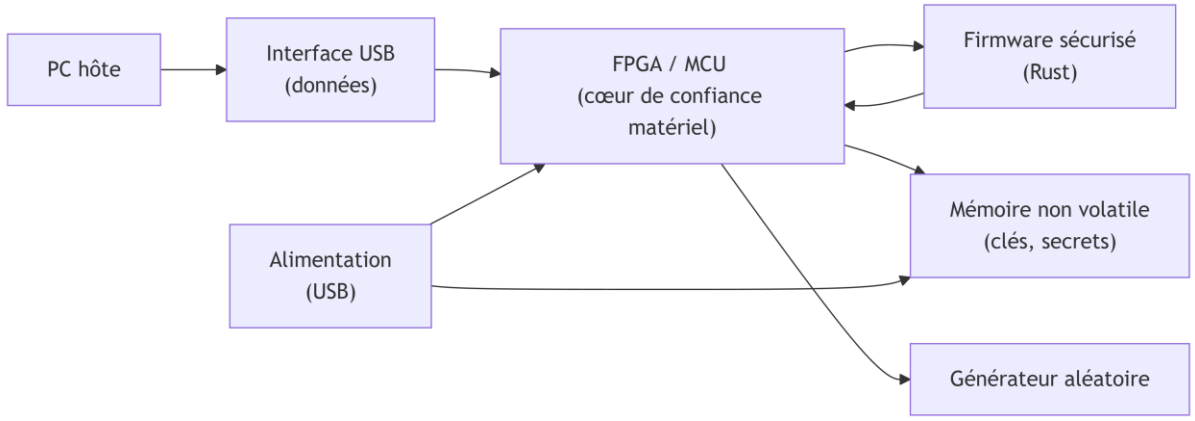
2. Présentation des solutions existantes

L'état de l'art des solutions d'authentification matérielle met en évidence un compromis entre niveau de sécurité, ouverture technologique et flexibilité d'intégration. Le tableau ci-dessous synthétise les principales solutions existantes et leurs limites.

Dans ce contexte, la solution Tillitis Key 1 se distingue par son approche open-hardware et open-firmware, offrant un cadre pertinent pour l'étude, la reproduction et l'amélioration d'un dispositif de sécurité matérielle.

Solution	Type	Open-source (HW / FW)	Niveau de sécurité	Limites principales
Tillitis Key 1 (TKey)	Clé matérielle basée FPGA	Oui / Oui	Élevé	Coût, fabrication complexe, nécessite compétences hardware
YubiKey	Clé matérielle USB	Non / Partiel	Très élevé	Solution propriétaire, peu de flexibilité, non modifiable
Nitrokey	Clé matérielle USB	Partiel / Oui	Élevé	Hardware partiellement fermé, performances limitées
Secure Element (ATECC, SE050…)	Circuit sécurisé dédié	Non / Non	Très élevé	Boîte noire, dépendance fournisseur, peu pédagogique
TPM (Trusted Platform Module)	Module matériel embarqué	Non / Non	Élevé	Intégration complexe, usage orienté plateforme plutôt que clé
Smartcard classique	Carte à puce	Non / Partiel	Moyen à élevé	Vulnérable à certaines attaques physiques, vieillissant
Solutions logicielles (2FA logiciel)	Authentification logicielle	Oui / Oui	Faible à moyen	Exposées aux malwares, dépendantes de l'OS

3. État de l'art sur l'architecture matérielle des dispositifs d'authentification



Les dispositifs d'authentification matériels, tels que les clés de sécurité USB, reposent sur une architecture compacte mais critique, dont la fiabilité conditionne directement le niveau de sécurité offert. Dans une approche open hardware, l'enjeu est de comprendre et maîtriser chaque brique matérielle afin de garantir à la fois transparence, reproductibilité et résistance aux attaques.

De manière générale, une clé d'authentification se compose d'une interface hôte, le plus souvent USB, d'un cœur de calcul chargé d'exécuter la logique de sécurité, d'un mécanisme de stockage ou de dérivation de secrets, ainsi que de composants périphériques essentiels comme les horloges, la gestion de l'alimentation et les sources d'entropie. L'ensemble doit tenir dans un facteur de forme réduit, tout en respectant des contraintes strictes de consommation et de robustesse électrique.

3.1. Cœur de calcul : FPGA et microcontrôleurs

Deux grandes approches architecturales coexistent dans l'écosystème des clés d'authentification open hardware : l'utilisation d'un microcontrôleur généraliste ou celle d'un FPGA.

Dans le cas de la TKey, le choix s'est porté sur un FPGA de la famille Lattice Semiconductor, plus précisément un iCE40 UltraPlus. Ce type de composant permet de concevoir une plateforme matérielle programmable intégrant un cœur processeur, des périphériques et des blocs cryptographiques directement en logique matérielle. L'intérêt principal réside dans la flexibilité offerte et dans la possibilité de mettre en œuvre un

modèle de type measured boot, où l'application exécutée est mesurée avant son lancement et influence directement la dérivation des secrets internes. Cette approche limite fortement l'impact d'une compromission logicielle, puisqu'un code modifié entraîne automatiquement une identité cryptographique différente.

À l'inverse, de nombreux projets similaires reposent sur des microcontrôleurs classiques, tels que les STM32, les nRF52 ou encore les ESP32. Les STM32, notamment, sont largement utilisés dans des clés FIDO2 open source en raison de leur maturité industrielle, de leur faible consommation et de la disponibilité de bibliothèques cryptographiques éprouvées. Les microcontrôleurs offrent une architecture plus simple à appréhender et à industrialiser, mais reposent davantage sur des mécanismes logiciels pour assurer l'isolation et la sécurité, ce qui peut augmenter la surface d'attaque si ces mécanismes sont mal implémentés.

Le choix entre FPGA et microcontrôleur dépend donc du compromis recherché entre flexibilité, transparence, complexité de développement et niveau de confiance accordé à la chaîne logicielle.

3.2. Secure elements et racine de confiance

Un autre élément clé de l'architecture matérielle est la gestion des secrets cryptographiques. De nombreux dispositifs intègrent un secure element, tel que l'ATECC608, conçu pour stocker des clés de manière isolée et exécuter des opérations cryptographiques sans exposer les secrets au reste du système. Ce type de composant apporte une résistance accrue face aux attaques physiques, notamment à l'extraction directe de clés.

Dans des architectures plus proches de plateformes informatiques, on peut également rencontrer des modules TPM (Trusted Platform Module), normalisés et conçus pour fournir une racine de confiance matérielle complète. Toutefois, leur intégration dans des clés USB compactes reste marginale en raison de leur coût, de leur encombrement et de leur complexité.

L'approche adoptée par la TKey se distingue en s'affranchissant d'un secure element dédié. La racine de confiance est construite à partir d'un secret propre au dispositif et d'un mécanisme de dérivation cryptographique dépendant de l'application mesurée. Cette philosophie, proche des concepts DICE, privilégie la transparence et la vérifiabilité du design, au prix d'une dépendance accrue à la qualité du durcissement matériel et logique de la plateforme.

3.3. Architecture PCB et composants critiques

La conception du circuit imprimé joue un rôle déterminant dans la sécurité globale du dispositif. Les clés d'authentification modernes utilisent généralement des PCB à deux ou quatre couches, ces dernières offrant de meilleures performances en termes de plans de masse, de stabilité de l'alimentation et de réduction des émissions électromagnétiques. Un routage soigné est indispensable, en particulier pour les lignes USB différentielles, qui doivent respecter des contraintes d'impédance et de symétrie afin d'assurer la fiabilité des communications.

L'alimentation constitue un autre point critique. Le passage de la tension USB vers les niveaux requis par le cœur logique impose l'utilisation de régulateurs adaptés, accompagnés d'un découplage efficace et de filtres permettant de limiter les perturbations. Une alimentation mal filtrée peut en effet faciliter certaines attaques par injection de fautes.

Enfin, les oscillateurs et les sources d'entropie sont essentiels au bon fonctionnement et à la sécurité du système. Les horloges conditionnent la stabilité des protocoles et sont une cible privilégiée pour les attaques par glitch. Les générateurs d'aléa, quant à eux, sont indispensables à la génération de clés et de nonces cryptographiques ; leur qualité impacte directement la robustesse des mécanismes de sécurité mis en œuvre.

4. État de l'art sur la sécurité hardware

La sécurité matérielle constitue un enjeu central pour les dispositifs d'authentification, dans la mesure où ces derniers sont destinés à être manipulés physiquement par des utilisateurs finaux et potentiellement accessibles à un attaquant. Contrairement aux systèmes purement logiciels, les clés d'authentification exposent directement leurs composants électroniques, ce qui ouvre la voie à des attaques exploitant les propriétés physiques du matériel, en complément des vulnérabilités logicielles classiques.

Menace / attaque	Description	Impact potentiel	Contre-mesures envisagées	Limites / remarques
Compromission du PC hôte	Le système hôte peut être infecté par un malware ou contrôlé par un attaquant	Vol ou manipulation des données échangées	Isolation stricte des secrets dans la clé matérielle ; aucune exposition des clés via USB	Ne protège pas contre une mauvaise utilisation légitime
Attaque par modification du firmware	Chargement ou exécution d'un	Dérivation ou exfiltration de secrets	*Measured boot* : les secrets dépendent de	Dépend de l'intégrité de la

	firmware modifié		l'application mesurée	chaîne de démarrage
Extraction de secrets en mémoire	Lecture directe de la mémoire non volatile	Compromission totale de l'identité	Accès contrôlé par le FPGA ; dérivation dynamique des secrets	Moins robuste qu'un secure element certifié
Side-channel (temps, consommation)	Analyse des variations de temps ou de puissance	Inférence d'informations sensibles	Implémentations cryptographiques constantes en temps ; réduction des fuites	Protection partielle sans contre-mesures dédiées
Fault injection (glitch)	Perturbation de l'alimentation ou de l'horloge	Contournement de contrôles de sécurité	Filtrage de l'alimentation ; surveillance des états internes	Contre-mesures limitées sur PCB simple
Attaques sur l'horloge	Désynchronisation volontaire du système	Comportements inattendus	Stabilité de l'oscillateur ; contrôles de cohérence	Détection difficile sans circuits dédiés
Faible entropie	Générateur aléatoire de mauvaise qualité	Clés prédictibles	Utilisation d'une source d'entropie matérielle	Qualité dépendante de l'implémentation
Attaque physique invasive	Accès direct au composant (sonde, décapsulation)	Compromission complète	Absence de protection invasive dédiée (choix open-hardware)	Accepté comme compromis pédagogique

5. Chaînes logicielles et firmware open hardware

Dans le cadre de l'étude des chaînes logicielles et firmware open hardware, cette section explore les pratiques sécurisées et reproductibles qui ancrent la robustesse des systèmes matériels, en s'appuyant sur des cas concrets tels que Tillitis Key1 ou SoloKeys. Ces chaînes, intégrant firmware, drivers, bibliothèques matérielles et protocoles de sécurité, reposent sur une approche modulaire et tiercielle : l'utilisation de bibliothèques dédiées comme u-boot ou OpenOCD pour la gestion embarquée, l'intégration de mécanismes de sécurité matérielle tels que le secure boot via ED25519

ou l’attestation matérielle du modèle DICE, ainsi que la mise en œuvre de pipelines CI/CD automatisés (ex. : GitLab CI) pour garantir la traçabilité et la validation rigoureuse des builds. Contrairement aux solutions closes (ex. : Secure Boot Apple), les projets open hardware valorisent la transparence et la flexibilité, permettant une auditabilité complète et une adaptation aux besoins communautaires, tout en confrontant des défis comme la fragmentation des formats (eMMC vs. SPI) ou le manque d’outils standardisés pour analyser les chaînes logicielles. Cependant, en alignant ces pratiques sur les phases clés du projet, notamment le développement Rust sécurisé (Phase 3), le renforcement des protections matérielles (Phase 4) et l’évaluation fonctionnelle (Phase 5), il est possible de renforcer la résilience des systèmes en réduisant les vulnérabilités via des tests d’intrusion réguliers et une attestation logicielle cohérente avec le modèle DICE. L’objectif reste de structurer des standards partagés (ex. : Open Firmware Standards) et de renforcer la collaboration internationale pour favoriser une écosystème open hardware résilient et transparent, en répondant aux exigences strictes de sécurité matérielle tout en garantissant la reproductibilité à grande échelle.

6. Méthodes de reproduction open hardware

La reproduction matérielle dans le domaine de l'open hardware s'appuie sur une démarche structurée reposant sur des fichiers de conception précis, tels que ceux générés par KiCad ou les gerbers, pour traduire sans ambiguïté les schémas électroniques en prototypes physiques. L'analyse rigoureuse du Bill of Materials (BOM) et la sélection des composants souvent réalisée via des plateformes comme Digi-Key ou Mouser sont essentielles pour garantir la conformité technique, surtout pour les éléments critiques comme les microcontrôleurs ou les circuits intégrés. La fabrication des PCB est généralement externalisée via des services industriels (JLCPCB, PCBWay) ou des plateformes communautaires (Aisler), avec une attention particulière portée aux paramètres de fabrication (épaisseur de cuivre, résistance thermique) pour optimiser la fiabilité et la reproductibilité à grande échelle.

Les techniques d'assemblage, adaptées aux contraintes de précision et de budget, varient entre la soudure manuelle minutieuse pour les prototypes de petite série, le reflow pour les connexions haute densité, et le pick-and-place automatisé pour une industrialisation plus évoluée. En parallèle, les tests de validation couvrant la continuité des circuits, la stabilité alimentaire et la communication USB sont réalisés avec des équipements de base (multimètres, oscilloscopes) pour identifier les défauts avant l'intégration finale. Ces méthodes, illustrées par des projets comme le Tillitis Key1, mettent en avant la collaboration communautaire pour surmonter des défis logistiques (variabilité des composants, gestion des sources d'approvisionnement) tout en garantissant une reproductibilité rigoureuse.

Cette approche méthodique, directement alignée avec la Phase 2 du projet (vérification des schémas et tests de reproductibilité), favorise la transparence entre concepteurs et fabricants, tout en renforçant la fiabilité des prototypes grâce à des standards partagés. En combinant expertise technique et partage d'expertise, la reproduction matérielle open hardware permet d'obtenir des solutions robustes et accessibles, adaptées aux besoins spécifiques des communautés techniques.

7. Limites et manques des solutions existantes

Bien que ces différentes clés de sécurité offrent chacune des avantages, elles présentent aussi des contraintes structurelles qui influencent la capacité de l'équipe à les reproduire, les étudier, ou à s'en inspirer pour concevoir une solution améliorée.

SoloKeys souffre de plusieurs limites. Son écosystème reste moins mature que celui de marques historiques comme Yubico, ce qui se traduit par un support plus restreint et une communauté réduite, compliquant la reproduction complète du produit ainsi que la création d'un environnement logiciel riche. Malgré son firmware open source, la plateforme dépend fortement du secure element ATECC608, qui impose le stockage permanent des clés privées dans une puce sécurisée opaque. Cette contrainte limite les usages expérimentaux et restreint la possibilité d'explorer des approches alternatives de gestion des secrets.

Les YubiKeys présentent d'autres contraintes importantes. Elles reposent sur un firmware propriétaire, empêchant toute vérification indépendante du code et réduisant la transparence, un point pourtant essentiel pour les chercheurs, les institutions ou les projets académiques. Leur nature fermée empêche la personnalisation ou l'adaptation fine aux besoins spécifiques d'un projet. Malgré leur robustesse, les modèles les plus complets restent coûteux, ce qui peut freiner les déploiements massifs ou constituer un obstacle pour des travaux exploratoires nécessitant de nombreuses unités.

La Tillitis Key adopte une approche intéressante mais comporte également plusieurs limites structurelles. Elle ne repose sur aucun secure element, ce qui réduit sa résistance aux attaques physiques et impose de faire confiance au stockage interne d'un FPGA, moins durci face à des adversaires sophistiqués. Son firmware est simple mais pas véritablement durci, et la qualité de l'entropie générée reste perfectible. L'utilisation impose de charger une application à chaque connexion, rendant l'expérience moins fluide que celle des SoloKeys ou YubiKeys. De plus, la plateforme n'est pas conçue comme un système modulaire pouvant s'intégrer à des contextes IoT, et son écosystème, encore jeune, dispose d'un support limité et d'une base d'utilisateurs restreinte.

Au-delà de leurs différences, toutes ces solutions présentent des limites communes : SoloKeys mise sur l'ouverture mais reste dépendante d'un secure element propriétaire ; Yubico offre une solution très aboutie mais totalement fermée et impossible à reproduire ; Tillitis propose une architecture flexible mais moins résistante physiquement et encore peu mature. Ces constats montrent qu'il existe un réel besoin pour une nouvelle approche qui associe une ouverture complète, un durcissement matériel et logiciel, une reproductibilité totale et une meilleure isolation des secrets sans recourir à un composant opaque. C'est dans cette direction que se situent les pistes d'amélioration que notre équipe souhaite développer.

8. Synthèse : lien entre l'état de l'art et le projet

La TKey constitue une référence centrale pour notre projet, car elle adopte une architecture radicalement différente des clés d'authentification traditionnelles. Contrairement aux tokens classiques, elle ne stocke aucune clé privée de manière permanente : les secrets sont générés dynamiquement grâce au mécanisme de measured boot. Reproduire une TKey permet donc d'explorer une approche plus moderne et entièrement transparente, à l'opposé des modèles classiques reposant sur un microcontrôleur et un stockage fermé.

L'intérêt d'étudier la TKey est renforcé par le fait qu'il s'agit d'une plateforme entièrement open source. L'accès complet au matériel, au firmware, à la logique interne et aux applications offre une visibilité totale sur le fonctionnement du token. Cette transparence facilite l'analyse des mécanismes cryptographiques, la compréhension fine de l'architecture interne, ainsi que la possibilité d'adapter ou d'améliorer chaque composant en fonction des besoins de notre propre prototype. Cela représente un avantage majeur pour reproduire une solution similaire tout en proposant des évolutions pertinentes.

Les limites des solutions existantes montrent clairement l'intérêt et la pertinence de notre démarche. Aucune clé sur le marché ne combine aujourd'hui open hardware, reproductibilité totale et modèle de sécurité avancé, tout en restant adaptée à des usages pédagogiques. La Tillitis Key fournit une base très intéressante, mais certains aspects peuvent être améliorés, notamment en termes de durcissement matériel et logiciel ou de modularité pour l'apprentissage.

Notre projet vise précisément à combiner reproduction, compréhension approfondie et renforcement de la sécurité. L'objectif est de proposer une plateforme ouverte, auditable et suffisamment robuste pour servir à la fois de prototype expérimental et de support pédagogique. À terme, le LabCyber pourra utiliser ce dispositif pour illustrer les principes de la sécurité matérielle, former les étudiants ou tester de nouveaux mécanismes. Le projet s'inscrit ainsi naturellement dans la continuité des travaux existants, tout en apportant une contribution originale et utile à la communauté.

Objectifs du projet :

Le projet vise à reproduire et sécuriser un dispositif hardware open-source existant, tel que la Tillitis Key1, afin de comprendre en profondeur sa conception électronique, son architecture de sécurité et son firmware, puis de proposer une version améliorée et durcie.

Dans un premier temps, l'équipe réalisera l'analyse matérielle complète du dispositif, en étudiant sa chaîne technologique (FPGA, microcontrôleur, éventuel Secure Element), son routing PCB, son alimentation et sa gestion cryptographique interne. Cette étape permettra d'identifier les choix d'architecture qui ont été faits, les zones critiques et les points d'attaque potentiels afin de mieux comprendre la conception de la clé.

Par la suite, l'objectif sera de reproduire fidèlement la carte en fabriquant plusieurs exemplaires du PCB open hardware, en sélectionnant les composants adaptés, en réalisant le montage, puis en validant la conformité fonctionnelle du matériel recréé.

L'étape suivante consistera à reprendre et adapter le firmware existant (Rust ou C selon le projet open-source), afin de comprendre son fonctionnement interne et d'intégrer des améliorations de sécurité.

Le projet inclura une démarche complète de documentation open hardware, comprenant les schémas, les fichiers PCB, la nomenclature, le firmware, les outils de test, ainsi que les retours d'expérience techniques destinés au LabCyber et à la communauté.

Enfin, ce travail permettra de fournir au laboratoire un support pédagogique réutilisable, sous la forme d'un kit de travaux pratiques et de démonstrations dédié à la sécurité hardware, permettant de sensibiliser aux risques et aux mécanismes de durcissement dans les systèmes embarqués modernes.

Problèmes/difficultés rencontrés et actions pour chercher des solutions :

Deux principales difficultés se sont présentées à nous au cours du lancement du projet PI^2.

Premièrement, nous avions au départ pour objectif de nous lancer sur la fabrication ambitieuse d'une série de microcontrôleur ESP32. Cependant, après avoir présenté l'idée à nos partenaires, ces derniers nous ont suggéré de nous réorienter sur un tout autre projet, car ce dernier était beaucoup trop compliqué à réaliser avec les moyens que nous disposons. Nous avons donc été réorientés vers un autre sujet, tout aussi intéressant pour nous : la conception d'une "Tillitis Key".

Deuxièmement, nous avons rencontré des difficultés avec l'attribution des projets. En effet, après notre première réunion d'équipe que nous avions effectuée rapidement, nous avons été répartis dans différents groupes. Après avoir cherché une solution pour garder ce projet qui nous tenait à cœur, nous avons finalement pu reprendre ce sujet mais avec plusieurs semaines de retard.

Résultats obtenus :

Une recherche approfondie sur les Tkey a été menée par l'équipe afin de comprendre leur rôle et leur utilisation typique, dans le but de mieux cerner les enjeux associés. Les rôles au sein de l'équipe ont ensuite été attribués. Enfin, une analyse de la Tillitis Key1 et de son architecture a été réalisée pour identifier les composants et les étapes nécessaires à sa reproduction.

Résultats attendus pour les 3 semaines suivantes :

Au cours des trois prochaines semaines, notre objectif principal est de finaliser et livrer l'expression de besoin, accompagnée de l'estimation des coûts et de la liste détaillée des composants et matériels nécessaires à la réalisation des prototypes de clés.

Cette livraison est planifiée pour la semaine du 15 décembre 2025, conformément aux échanges précédents avec notre partenaire.

Ce jalon constitue un livrable majeur avant la clôture de l'année, car il permettra de valider les besoins techniques, de lancer les approvisionnements et de préparer la phase de reproduction matérielle dès le début de l'année suivante.

Gestion de projet :

Notre gestion de projet va s'appuyer sur une organisation claire des rôles de responsables et une définition des tâches.

Le travail est réparti entre les membres selon leurs compétences principales : un responsable hardware, un responsable firmware, un responsable sécurité et un chef de projet chargée de la coordination générale. Cette organisation permet d'allier spécialisation technique et vision transversale.

En termes de communication, nous avons un canal de discussion sur WhatsApp pour l'organisation des réunions, les questions spontanées, etc., et un canal Teams afin de

communiquer avec les tuteurs et déposer tous les documents demandés ainsi que les ressources externes utilisées.

L'objectif est d'avoir des points synchronisés pour suivre l'avancée des tâches une fois les réalisations techniques entamées. Un espace collaboratif partagé est également mis en place pour l'aspect technique (GitHub + Drive/Notion), qui organise les fichiers sources, rapports intermédiaires, schémas et protocoles de test afin d'éviter toute perte d'information.

Les décisions techniques importantes (choix des composants, modèle de Secure Boot, architecture firmware) seront systématiquement documentées pour assurer la traçabilité.

La communication avec les partenaires externes, tels que les fabricants de PCB ou la communauté Tillitis, se fera éventuellement si nécessaire, via des échanges formalisés (emails, tickets GitHub), afin d'assurer une bonne traçabilité et d'intégrer les retours techniques pertinents au projet.

Compétences utilisées (avec des exemples concrets pour illustrer ces compétences) :

A ce stade du projet, nous avons eu l'occasion de mobiliser et de développer plusieurs compétences clés, tant techniques que méthodologiques.

La compétence la plus sollicité jusqu'à présent a été la recherche documentaire, associé à un esprit de synthèse et d'analyse critique. Nous avons en effet exploré de nombreuses sources d'information et projets open hardware, notamment autour des dispositis de type TKey et ESP32, afin de comprendre leurs architectures, leurs usages et leurs limites.
Cette phase nous a permis de renforcer notre capacité à identifier des sources fiables, à comparer des solutions techniques et à formuler une vision claire du projet.

Nous avons également mis en avant notre résilience face aux différents changements rencontrés en début de projet (modification du sujet, réorganisation des équipes). Malgré ces ajustements, nous avons su maintenir notre motivation, faire preuve de souplesse et recentrer nos efforts sur un objectif commun et réalisable.

Enfin, la curiosité a été un atout essentiel. Face à un sujet initialement très ouvert, nous avons fait preuve d'initiative et d'exploration, en recherchant différents dispositifs open hardware et en envisageant plusieurs scénarios d'application. Cette démarche proactive nous a permis de mieux définir le périmètre technique et les enjeux de cybersécurité du projet.

A travers ces premières étapes, nous avons donc renforcé nos compétences d'investigation, de communication et d'adaptation, qui constituent une base solide pour aborder la phase de conception technique à venir.

Rôles au sein de l'équipe :

Pour commencer la présentation de l'équipe, nous avons **Paul ARTISI**, référent du groupe pour ce projet. Alternant chez **SFR** en gouvernance cybersécurité, il manipule déjà ce type d'outils, notamment la solution Yubikey. Étant arrivé après la mise en place du projet, son expérience porte principalement sur la conduite du changement et sur l'application de cette norme au sein des différentes entités de l'entreprise. Dans le cadre du projet, il occupera le rôle d'intermédiaire principal avec les tuteurs entreprise/école, ainsi que les missions d'organisation, de gestion d'équipe et d'élaboration du planning et des livrables associés.

Le second membre de l'équipe est **Aboubakar BAOUCHI**, alternant chez **IBM** en avant-vente cloud. Il apportera au projet son expertise dans la définition des attentes et des objectifs clés à anticiper pour le partenaire. Son intérêt marqué pour la technique fait de lui un atout majeur. Il aura ainsi un rôle plutôt technique et supervisera l'ensemble des jalons liés à cette dimension.

Le troisième membre de l'équipe, **Fanantenana Michael ANDRIANIRINA** est alternant chez **XLConcept** en tant que développeur full-stack. En plus du développement, il réalise notamment des audits sur les applications créées au sein de l'entreprise, ce qui lui apporte une expérience précieuse en analyse technique et en bonnes pratiques de sécurité. Ses compétences lui permettront de contribuer efficacement à la réalisation de l'état de l'art, ainsi qu'à la production des documents techniques et méthodologiques du projet. Son rôle au sein de l'équipe sera donc fortement orienté vers l'aspect technique, en cohérence avec son expertise et ses objectifs de montée en compétences.

La quatrième membre de notre équipe est **Eliza KARA,** alternante chez **Bouygues Telecom**. L'année dernière, elle a travaillé sur des missions de gouvernance ainsi que sur la gestion du risque cyber pour l'ensemble des fournisseurs. Cette année, elle a rejoint l'équipe de développement des box internet, où elle contribue à des projets orientés sécurité. Grâce à son expérience variée et à ses compétences techniques : notamment sur l'outil KiCad, elle jouera un rôle clé dans la réalisation de l'état de l'art et apportera un soutien essentiel sur les aspects techniques du projet.

Le cinquième membre de l'équipe est **Nathan HERBRON**, il travaille chez **SYSGO France SAS**, une filiale de THALES spécialisée dans la vente d'un système d'exploitation sécurisé dédié à l'embarqué, principalement pour les secteurs de l'aviation et de la défense, avec une expansion récente vers l'automobile. Il occupe un rôle de *lab admin*, où il assure la maintenance et la gestion du laboratoire de test. Il réalise également du développement de tests logiciels bas niveau et sera amené, au cours de l'année, à

développer des BSP (Board Support Packages). Son expertise dans l'embarqué et les environnements critiques apportera une forte valeur ajoutée au projet, notamment sur les aspects techniques avancés et sur l'analyse des besoins en sécurité.

Le sixième membre de notre équipe est **Sofien EL BAHI,** il est alternant chez **1-more-thing,** une TPE spécialisée dans le cloud. Il occupe un rôle orienté gouvernance et pilote la mise en place de la cybersécurité nécessaire à l'obtention de la certification ISO 27001. Parallèlement, il possède un fort intérêt pour l'expérimentation, notamment sur les aspects hardware et les solutions open source. Ces appétences font de lui un atout pertinent pour le projet, en particulier sur les volets techniques liés au matériel et aux outils ouverts.

Planing prévisionnel :

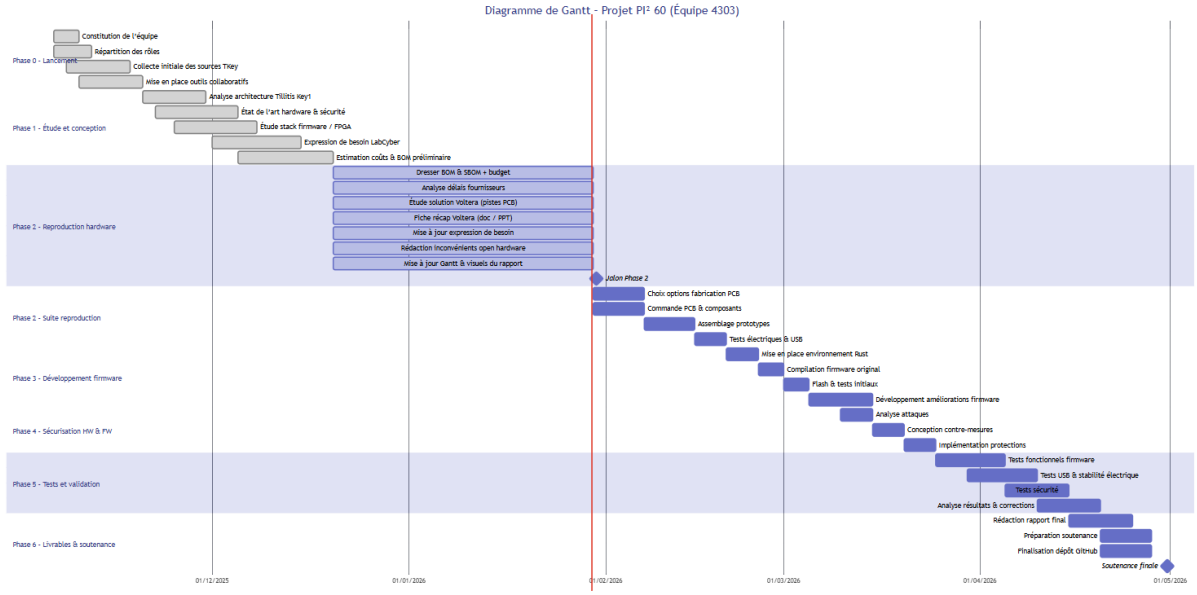
Phase	Objectif	Sous-tâches identifiées
Phase 1 Etude et conception	Comprendre le design existant et définir les besoins du Lab	-Analyse de Tillitis Key1 et de son architecture. -Étude de la stack : hardware, firmware, bootloader, outils de build. -Analyse des options de fabrication (PCB, composants, coûts). -Analyse des risques sécurité (physiques & logiques).
Phase 2 Reproduction	Reproduire le TKey	-Récupération des fichiers KiCad/gerbers du projet open-source. -Vérification des BOM et sourcing des composants. -Fabrication PCB (JLCPCB, Aisler...). -Montage manuel ou assemblage. -Test de continuité & alimentation.
Phase 3 Développement firmware	Développer / adapter le firmware Iot	-Compilation de la version open-source. -Mise en place des toolchains (Rust, cargo, scripts). -Flash et premiers tests sur le hardware reproduit.

		-Développement des améliorations prévues (nouveau protocole, signatures, random seed, etc.).
Phase 4 Sécurisation	Ajouter les protections de cybersécurité	-Analyse des attaques possibles -Mise en place de protections (ex. : masquage des opérations critiques) -Implémentations et tests (simulations d'attaques avec des outils spécifiques aux FPGA)
Phase 5 Tests et validation	Vérifier la robustesse et documenter les résultats	-Tests fonctionnels et d'intégrité du firmware (validation des protocoles d'authentification dans un environnement contrôlé). -Rédaction du bilan technique
Phase 6 Livrable et partage	Diffuser et valoriser le projet	-Documentation globale - Repo GitHub complet (code, PCB, docs) -Présentation finale -élaboration du rapport final

Retroplanning :

Gantt (Simplifié)

Gantt (Détaillé)



Bibliographie :

Sources générales :

- https://olimex.com/Products/IoT/ESP32-S3/ESP32-S3-DevKit-Lipo/open-source-hardware
- https://oshwlab.com/Zulfahmi27/esp32-devkit-v1

TKey :

- https://www.tillitis.se/
- https://fr.hwlibre.com/S%C3%A9curit%C3%A9-des-appareils-avec-Zephyr-RTOS%C2%A0%3A-tout-ce-qui-compte/
- https://github.com/tillitis/tillitis-key1/
- https://github.com/tillitis/tk1-pcba/tree/main

Pour l'état de l'art :

Tillitis Key1 :

- Site officiel Tillitis : https://www.tillitis.se
- Dépôt GitHub TK1 : https://github.com/tillitis/tillitis-key1
- Repository TK1-PCBA (hardware) : https://github.com/tillitis/tk1-pcba

SoloKeys :

- Site officiel : https://solokeys.com
- Dépôts GitHub SoloKeys : https://github.com/solokeys

Nitrokey :

- Site officiel Nitrokey : https://www.nitrokey.com
- Nitrokey FIDO2 : https://www.nitrokey.com/products/nitrokey-fido2

OnlyKey :

- Site officiel : https://www.onlykey.io
- Documentation firmware : https://docs.crp.to

YubiKey :

- Site Yubico : https://www.yubico.com
- Documentation WebAuthn / FIDO2 : https://developers.yubico.com