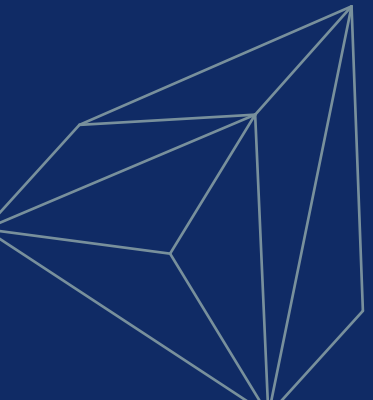
CIA

Fondations de la Communication Sécurisée

Loïc Rouquette

Sommaire

Introduction



Hyper-connectivité au service de l'Industriels

- L'IIoT (Internet des Objets Industriels) est la convergence de l'IT et de l'OT.
- Objectif : optimiser les processus et la collecte de données en temps réel.
- Secteurs d'application : services publics, fabrication, sante IoMT (Internet des Objets Médicaux), etc.
- Résultat : des milliards d'appareils connectés.

Une surface d'attaque en expansion

- Hyper-connectivité crée une surface d'attaque sans précédent.
- **Pourquoi les appareils IIoT sont-ils vulnérables ?**
  - Déploiement en grand nombre, souvent dans des environnements non sécurisés.
  - Ressources limitées (calcul, mémoire) : impossibilité d'installer des logiciels de sécurité traditionnels (ex : EDR (Endpoint Detection and Response), anti-virus, etc.)
  - Connexion entre systèmes hérités (ICS, SCADA) et des réseaux IP modernes.

Sécurisé l'IIoT : un défi de taille

- Les systèmes hérites n'ont pas été conçus pour la sécurité en ligne.
- L'IIoT crére de "nouveaux et dangereux vecteurs d'attaque" entre ces anciens systèmes et les réseaux modernes.

Conclusion : La sécurité ne peut plus être une réflexion après coup. Elle doit être intégrée dès la conception des solutions IIoT.

Au délà des vol de données : Les nouvelles menaces IIoT

- Les attaques IIoT ne visent plus seulement le vol de données.
- L'objectif est désormais le **sabotage**, la **perturbation** et la **mise en danger physique**.
- Un simple capteur peut devenir le point de départ d'une catastrophe.

Quatres types d'attaques sur des systèmes l'IIoT

- **Mouvement latéral** : Un attaquant compromet un appareil simple (caméra, capteur) pour s'infiltrer dans le résqeau et cibler des systèmes crutuqyes comme les PLC (Programmable Logic Controller).
- **Botnets massifs** : Les appareils IIoT non sécurisés sont utilisés pour créer des vastes armées de "zombies" pour des attaques DDoS, comme dans le cas d'une université (anonymisée) dont 5000 appareils ont été compromis.
- **Sabotage physique** : Les attauqes manipulent le monde physique. Exemples : couper le chauffage d'immeubles en hiver (Filande) ou désactiver des dispositifs médicaux vitaux (pompes à insuline).
- **Attaques sur la chaîne d'approvisionnement** : Un composant compromis dès la fabrication devient un vecteur d'attaque une fois déployé dans une usine.

Du risque cyber au risque physique

- La convergence entre l'IT et l'OT transforme la nature du risque.
- **IT traditionnel** : fuite de données, perte financière.
- **OT et IIoT** : exploision d'usines, contamination de l'eau, accidents mortels.
- Les menaces de cybersécurités ne sont plus virtuelles, elles ont directement et potentiellement **catastrophiques**.

Les trois piliers de la sécurité

Les trois piliers de la sécurité

Confidentialité

Les trois piliers de la sécurité

Confidentialité

Il ne s'agit pas seulement de protéger des sécrets mais d'empêcher un attaquant de modifier des commandes de contrôle.

Les trois piliers de la sécurité

Confidentialité

Il ne s'agit pas seulement de protéger des sécrets mais d'empêcher un attaquant de modifier des commandes de contrôle.

Intégrité

Les trois piliers de la sécurité

Confidentialité

Il ne s'agit pas seulement de protéger des sécrets mais d'empêcher un attaquant de modifier des commandes de contrôle.

Intégrité

Le pilier le plus critique pour l'IIoT. Il faut garantir que les données et commandes ne sont pas altérées.

Les trois piliers de la sécurité

Confidentialité

Il ne s'agit pas seulement de protéger des sécrets mais d'empêcher un attaquant de modifier des commandes de contrôle.

Intégrité

Le pilier le plus critique pour l'IIoT. Il faut garantir que les données et commandes ne sont pas altérées.

Disponibilité

Les trois piliers de la sécurité

Confidentialité

Il ne s'agit pas seulement de protéger des sécrets mais d'empêcher un attaquant de modifier des commandes de contrôle.

Intégrité

Le pilier le plus critique pour l'IIoT. Il faut garantir que les données et commandes ne sont pas altérées.

Disponibilité

Garantir que les systèmes critiques restent opérationnels. Une attaque DDoS peut rendre une usine "aveugle" et dangereuse.

Le protocole TLS : une fondation solide

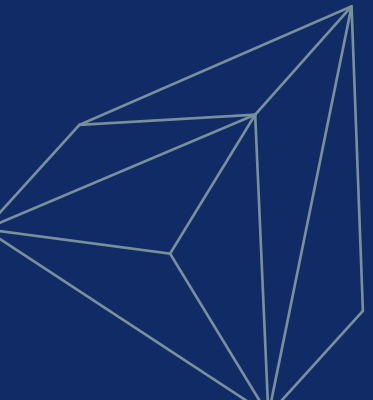
- Pour sécurisé les communications, le protocole TLS (Transport Layer Security) a été standardisé.
- Il garantit la **confidentialité**, l'**intégrité** et l'**authentification**.
- Comprendre le protocole TLS est essentiel pour sécuriser les systèmes IIoT.

Pour construire un canal de communication sécurisé, deux éléments fondamentaux sont nécessaires :

- un mécanisme pour établir la confiance entre les parties communicantes ;
- un ensemble cryptographiques pour protéger les données échangées.

Le premier est réalisé par une PKI, le second par choix judicieux de primitives cryptographiques.

Infrastructure à Clé Publique (PKI)



Rappels

Qu’est ce que la cryptographie à clé symétrique ?

Rappels

Qu'est ce que la cryptographie à clé symétrique ? La cryptographie à clé symétrique est une méthode de chiffrement où la même clé secrète est utilisée pour chiffrer et déchiffrer les données.

Elle assure la confidentialité des communications.

Rappels

Qu'est ce que la cryptographie à clé symétrique ? La cryptographie à clé symétrique est une méthode de chiffrement où la même clé secrète est utilisée pour chiffrer et déchiffrer les données.

Elle assure la confidentialité des communications.

Qu'est ce que la cryptographie à clés asymétriques ?

Rappels

Qu'est ce que la cryptographie à clé symétrique ? La cryptographie à clé symétrique est une méthode de chiffrement où la même clé secrète est utilisée pour chiffrer et déchiffrer les données.

Elle assure la confidentialité des communications.

Qu'est ce que la cryptographie à clés asymétriques ? La cryptographie à clés asymétriques utilise une paire de clés mathématiquement liées, une clé **publique** pour le **chiffrement** et une clé **privée** pour le **déchiffrement**.

Elle assure l'authentification d'une entité connue.

Rappels

Qu'est ce que la cryptographie à clé symétrique ? La cryptographie à clé symétrique est une méthode de chiffrement où la même clé secrète est utilisée pour chiffrer et déchiffrer les données.

Elle assure la confidentialité des communications.

Qu'est ce que la cryptographie à clés asymétriques ? La cryptographie à clés asymétriques utilise une paire de clés mathématiquement liées, une clé **publique** pour le **chiffrement** et une clé **privée** pour le **déchiffrement**.

Elle assure l'authentification d'une entité connue.

Qu'est ce qui assure l'identification d'une entité inconnue ?

PKI : Une chaîne de confiance pour Internet

- La PKI est un système de gestion des clés publiques.
- Elle repose sur une CA (Autorité de Certification), une tierce partie de confiance.
- Le principe : on ne fait pas confiance à toutes les clés, mais à un petit nombre de CA racines.
- Ces CA signent des **certificats numériques** qui lient une identité (un nom de domaine, un appareil IIoT, etc.) à une clé publique.
- Schéma de la chaîne de confiance :
  - CA racine : Le sommet de la hiérarchie. Sa clé privé est ultra-protégée (matériel spécifique). Elle ne signe des certificats que pour des CA intermédiaires.
  - CA intermédiaires : Déléguent le pouvoir de signature de la CA racine. Elles signent des certificats pour des entités finales.
  - Entités finales : Sites web, appareils IIoT, etc. Elles utilisent les certificats pour prouver leur identité.

Le rôle des CA

- Le rôle d'une CA est de **valider l'identité du demandeur** avant de délivrer un certificat. Ce processus est audité rigoureusement.
- La CA gère aussi le **cycle de vie** des certificats :
  - Émission ;
  - Révocation : Publication des **Listes de Révocation de Certificats** (CRL) si une clé privée est compromise ;
- C'est cette rigueur qui assure la confiance dans l'écosystème.

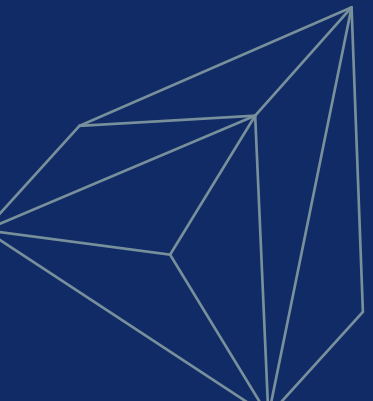
Anatomie d'un certificat X.509 v3

Champ / Extension	Description	Importance et Pertinance
Issuer	Nom unique (DN) de l'autorité de Certification qui a signé le certificat.	Ancre la confiance dans la chaîne de certification.
Subject	Nom unique (DN) de l'entité (serveur, appareil) identifiée par le certificat.	Identifie le détenteur du certificat.
Validity (Not Before / Not After)	Période de validité du certificat.	Empêche l'utilisation de certificats expirés ou pas encore valides.
Subject Public Key Info	Clé publique de l'entité et algorithme associé (ex. ECC, RSA).	C'est l'élément central utilisé pour la cryptographie (vérification de signature, chiffrement).

Anatomie d'un certificat X.509 v3

Champ / Extension	Description	Importance et Pertinance
Subject Alternative Name (SAN)	Liste d'identifiants alternatifs (noms DNS, adresses IP).	**Champ critique** pour la validation du nom d'hôte. Remplace le cham Common Name (CN) déprécié.
Key Usage	Définit les opérations cryptographiques autorisées pour la clé (ex. : digitalSignature).	Restreint l'utilisation de la clé pour prévenir des abus.
Extended Key Usage (EKU)	Définit les applications autorisées pour la clé (ex. : serverAuth, clientAuth).	Spécifie le rôle du certificat dans un protocole comme TLS.

Primitives Cryptographiques Modernes



Sécurisé la communication : le rôle de TLS

- Une fois la confiance établie par la PKI, le protocole TLS sécurise la communication elle-même.
- TLS ne se limite pas à un seul algorithme ; il permet au client et au serveur de **négocier** une **suite de chiffrement** (cipher suite).
- Une suite de chiffrement est une **combinaison d'algorithmes** travaillant ensemble.

Les composantes d'une suite de chiffrement

`TLS_AES_128_GCM_SHA256` (TLS 1.3)

- `AES_128_GCM` : Algorithme de chiffrement symétrique (AES) avec une clé de 128 bits. Galois / Counter Mode (GCM) pour la confidentialité et l'intégrité.
- `SHA256` : Algorithme de hachage (SHA-256) utilisé dans le mode AEAD pour l'authentification des messages et pour le calcul de clés ddans le protocole de dérivation des clés.

Il est important de noter que dans TLS 1.3, l'algorithme d'échange de clés (ex. ECDHE) et l'algorithme de signature (ex. RSA, ECDSA) sont négociés séparément et ne font plus partie de la suite de chiffrement.

Les opérations coûteuses : Échange de clés et signatures

- L'**échange de clés** et la **signature** sont les opérations les plus gourmandes en calcul.
- Elles reposent sur la cryptographie asymétrique.
- Historiquement, l'algorithme **RSA** a dominé. Il est basé sur la difficulté des factoriser de très grands nombres.

RSA : Puissance et limitations

- Pour rester sécurisé, la taille des clés RSA a dû augmenté avec le temps (de 1024 bits à 2048 bits, voire 4096 bits).
- **Le problème** : des clés plus longues impliquent des calculs **exponentiellement plus lents** et une **consommation d'énergie** accrue.
- Cela rend le RSA peu adapté aux appareils IIoT, qui ont des ressources limitées.

L'ECC (Cryptographie à Courbes Elliptiques) : La solution pour l'IIoT

- La Cryptographie à Courbes Elliptiques (ECC) offre une solution bien plus efficace.
- **Le grand avantage** : Elle offre le **même niveau de sécurité que RSA avec des clés beaucoup plus courtes**.
- Par exemple, une clé ECC de 256 bits offre une sécurité comparable à une clé RSA de 3072 bits.
- Pour l'IIoT, cela se traduit part :
  - Des calculs plus rapide (via **ECDHE** ou **ECDSA**) ;
  - Une consommation d'énergie réduite ;
  - Moins de données à transmettre (certificats et signatures plus petits).

Échange de Clés et Signatures : RSA vs ECDHE/ECDSA

- RSA, robuste mais gourmand, est délaissé au profit de l'ECC dans le monde des objets connectés.
- L'ECC est la solution de choix pour les appareils IIoT en raison de son efficacité et de sa sécurité.

Sécurisé la communication : Le chiffrement symétrique

- Une fois les clés échangées, la communication utilise un **chiffrement symétrique**.
- C'est une opération beaucoup **plus rapide** que la cryptographie asymétrique.
- Les algorithmes modernes combinent chiffrement et authentification en une seule étape. On les appelle AEAD (Authenticated Encryption with Associated Data).
- Ce procédé garantit la **confidentialité** et l'**intégrité** des données simulatnément.

Chiffrement Symétrique Authentifié : AES-GCM vs ChaCha20-Poly1305

- Le choix de l'algorithme dépend principalement de l'appareil. Les deux principaux sont :
  - `AES-GCM`
    - La **norme de facto**
    - **Extrêmement performant sur les processeurs modernes** sur les processeurs modernes (Intel, AMD, certains ARM)
    - Il bénéficie d'une **accélération matérielle** (instructions AES-NI) qui le rend très rapide et efficace.
  - `ChaCha20-Poly1305`
    - Une alternativer **plus récente**
    - **Excellente pêrformance logicielle** sans accélération matérielle
    - Il est souvent **plus rapide et moins énergivore** que l'`AES_GCM` sur les microcontrôleurs et les processuers **ARM bas de gamme**.

Chiffrement Symétrique Authentifié : AES-GCM vs ChaCha20-Poly1305

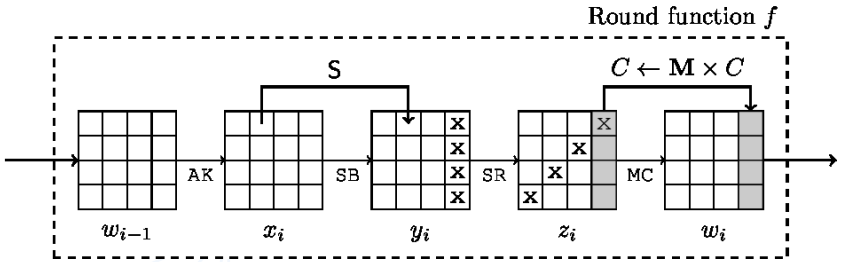


Figure 1: AES Round Function.

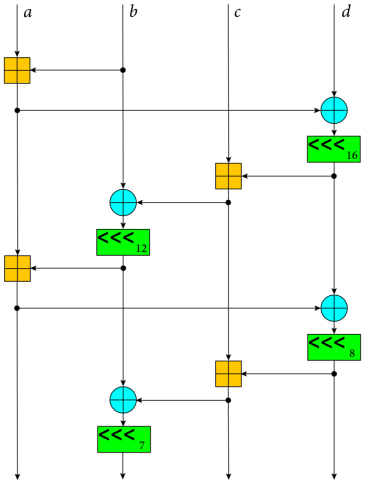
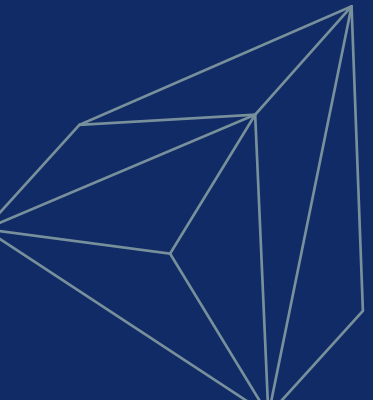


Figure 2: ChaCha Quarter Round Function.

Tableau récapitulatif des Algorithmes Cryptographiques pour l'IIoT

Cat.	Algorithme	Taille de clé	Performance CPU (sans HW)	Consommation énergétique	Impact Bande Passante	Dépendance Matérielle
Échange de Clé / Signature	RSA	3072 bits	Lent	Élevée	Élevé (certificats/ signatures volumineux)	Aucune
Échange de Clé / Signature	ECDSA/ECDHE	256 bits	Rapide	Faible	Faible	Aucune
Chiffrement AEAD	`AES_256_GCM`	256 bits	Lente	Modérée	Faible	**Optimale avec AES-NI**
Chiffrement AEAD	`ChaCha20_Poly1305`	256 bits	Très Rapide	Faible	Faible	Aucune (performant en pur logiciel)

Plongée en Profondeur dans TLS 1.3



Le protocole Transport Layer Security (TLS) est l'aboutissement de décennies de recherche de d'ingénieurie en sécurité des réseaux. Sa dernière version majeure, **TLS 1.3**, standardisée en **2018** dans la **RFC 8446**, représente une refonte significative visant à améliorer à la fois la sécurité et les performances par rapport à ses prédécesseurs.

Évolution de TLS : de la complexité à la simplicité

- Le pssage de TLS 1.3 à TLS 1.2 n'est pas un simple mise à jour, mais un changement philosophique profond.
- **TLS 1.2 était complexe** : beaucoup d'options, de suite de chiffrement, ce qui créait des failles de sécurité.
- Les attaquants pouvaint utiliser des **attaques par repli** (downgrade attacks) pour forcer l'utilisation de versions ou d'algorithmes plus faibles.
- TLS 1.3 adapte une approche de **"sécurité opinionée"** : on supprime les options pour ne garder que les meilleures.
- La simplicité et la sécurité par défaut priment sur la flexibilité.

TLS 1.3 vs. TLS 1.2 : Les différences clés

- **Latence du Handshake** : La plus grande amélioration. On passe de **2 allers-retours à 1 aller-retour** (1-RTT). Cela divise par deux le temps de connexion, crucial pour la performance.
- **Confidentialité Persistante (PFS)** : Rendue **obligatoire** en TLS 1.3. Cela garanti que si la clé privée du serveur est compromise, les sessions passées ne peuvent pas être déchiffrées.
- **Choix des Algorithmes** : TLS 1.3 élimine les algorithmes faibles et ne garde que **5 suites AEAD**, toutes robutes.

Simplicité, Robustesse et Performance

- **Suppression des vulnérabilités** : TLS 1.3 élimine des algorithmes obsolètes (RSA, RC4, 3DES, SHA-1, etc.). Cela supprime des classes entières de vulnérabilités.
- **Handshake confidentiel** : Le certificat du serveur n'est plus envoyé en clair, mais chiffré durant le *handshake*. Cella protège d'avantage la communication.
- **Leçon à retenir** : La meilleure façon de sécuriser n'est pas forcement d'ajouter des fonctionnalites, mais de **réduire la surface d'attaque**.

L’avenir de la séucrité des communications

- TLS 1.3 marque un tournant en ingénieurie de la sécurite.
- Il offre une **sécurité supérieure**, une **meilleure performance** et une **configuration plus simple**.
- Il devient la norme pour sécuriser les communications sur Internet et dans l’IIoT.

Handshake TLS 1.3 : Un échange en 1-RTT

- Le génie de **TLS 1.3** est de réduire le temps d'établissement d'une connexion sécurisée en un seul aller-retour (**1-RTT**).
- La clé est un restructuration complète du processus de handshake.
- **Le processus en bref** : Le client envoie un message avec tout ce dont le serveur a besoin. Le serveur répond en envoyant la suite de la communication déjà chiffrée.

Étape 1 : `ClientHello` (le pari optimiste)

- Le client envoie une liste de ses capacités (versions TLS, suites de chiffrement, algorithmes de signature).
- **L'innovation clé** : Il inclut une clé publique éphémère (extension `key_share`). C'est un pari sur le groupe de clés que le serveur va choisir.

Étape 2 : `ServerHello`

- Le serveur choisit une suite de chiffrement et un groupe de clés parmi ceux proposés.
- Il génère sa propre paire de clés éphémères et renvoie sa clé publique dans le `ServerHello`.
- Le client et le serveur ont maintenant chacun la clé publique de l'autre. Ils peuvent calculer indépendamment un secret partagé en utilisant l'algorithme **ECDHE**.
- Ce secret est ensuite utilise par la fonction HKDF (HMAC Key Derivation Function) pour dérifer plusieurs clés secrètes distinctes, dédiées à des usages précis (chiffrement des données, authentification des messages, etc.).

Étape 3 : Dérivation des Clés

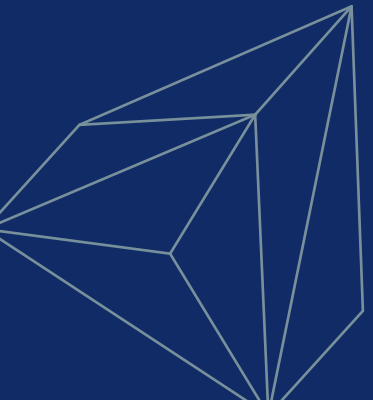
C'est une amélioration majeure par rapport à TLS 1.2, le serveur commence immédiatement à chiffrer ses messages.

- Le serveur envoie une série de messages chiffrés :
  - `EncryptedExtensions` : Contient des informations additionnnelles.
  - `Certificate` : Le serveur envoie sa chaîne de certificats. Ce message est maintenant chiffré pour préserver la confidentialité.
  - `CertificateVerify` : Le serveur prouve son identité en signant une transaction du *handshake* avec sa clé privée. Le client utilisera la clé publique pour valider cette signature.
  - `Finished` : Un MAC (Message Authentication Code) qui confirme que le *handshake* est intègre du côté serveur.

Étape 4 : Finished

- Le client reçoit les messages du serveur.
- Il déchifre les messages grâce aux clés dérivées précédemment.
- Il effectue les vérifications cruciales :
  - Valider le certificat du serveur.
  - Vérifier la signature dans `CertificateVerify`.
  - Vérifier le `Finished` du serveur.
- Si tout est correct, le client envoie à son tour son propre message `Finished` **chiffré**.
- Une fois ce message validé par le serveur, le **canal sécurisé est établi**. Le client et le serveur peuvent commencer à échanger des données applicatives chiffrées et authentifiées.

Conclusion



Sécurité IIoT : Du “pourquoi” au “comment”

- La sécurité dans l’IIoT est une nécessité absolue, pas une option.
- Comprendre les fondations de la communication sécurisée est crucial pour protéger les systèmes critiques.
- La sécurité de l’IIoT est une question d’**ingénieurie** et d’**optimisation**.
- Le choix d’algorithmes adaptés, comme **ECDSA** et **ChaCha20-Poly1305**, est essentiel pour concilier sécurité et performance.

TLS 1.3 : une réponse ciblée aux menaces IIoT

- **Authentification** (via X.509) : Prévient les attaques d'usurpation d'identité et de type "Man-in-the-Middle".
- **Confidentialité persistante** (via ECDHE) : Garantit que les communications passées ne peuvent pas être déchiffrées, même si une clé est compromise ultérieurement.
- **Intégrité et confidentialité** (via AEAD) : Assure que les données ne sont pas altérées ni espionnées pendant leur transit.

La transparence : une propriété de la sécurité

- La sécurité ne dépend pas seulement d'un protocole, mais aussi de sa **mise en œuvre**.
- La **transparence** est un élément de sécurité fondamental.
- Le fait que les norme comme TLS soient **ouvertes (RFC)** et implémentées dans des logiciels **open source (OpenSSL)** permet un audit et une vérification indépendants.

Questions ?