Micrologiciel HSM optimisé pour systèmes Autosar : la cryptographie sans limite pour l'automobile

La complexité croissante des logiciels et de la connectivité embarqués dans l'automobile requiert une protection cryptographique toujours plus importante. Cette protection doit également être implémentée par les systèmes Autosar en temps réel classiques. Les modules de sécurité matériels (Hardware Security Module - HSM) associés au micrologiciel approprié permettent d'utiliser une cryptographie à l'épreuve du temps, même lorsque les ressources sont rares.

Dans les automobiles, le niveau de connectivité des unités de contrôle électroniques ECU augmente depuis des années : elles sont connectées à la fois entre elles et avec le monde extérieur. Toutefois, comme les logiciels de plus en plus complexes embarqués sur ces ECU sont confrontés à de nouvelles exigences, les exigences en matière de communication augmentent également. Du point de vue de la sécurité, il s'agit de passer de systèmes uniques isolés à des nœuds hautement connectés. Par conséquent, la sécurité et la protection contre les menaces extérieures gagnent en importance. Cette protection peut notamment être assurée en augmentant le recours à la cryptographie forte – ce qui, à première vue, n'est pas facile dans les systèmes temps réel classiques.

Autrefois, la cryptographie n'était nécessaire que dans des modes de fonctionnement spécifiques tels que les mises à jour logicielles de l'ECU effectuées chez le garagiste. Aujourd'hui, elle doit pouvoir être utilisée de façon efficace même pendant l'exécution normale, par exemple pour authentifier les partenaires et les contenus de communication et empêcher les interceptions. L'implémentation appropriée doit être adaptée aux exigences temps réel. A première vue, cela contredit les exigences de temps de calcul des méthodes cryptographiques. Les modules de sécurité matériels (HSM) permettent néanmoins de résoudre ce problème, en affectant le calcul de la cryptographie à un processeur séparé. Cependant, ils nécessitent une implémentation optimisée pour atteindre leur plein potentiel.

AUTEUR

Tobias Jordan, expert des systèmes d'exploitation et de la sécurité chez Elektrobit Automotive.

Temps réel automobile

Les exigences associées au temps réel embarqué résultent généralement d'une chaîne d'effets qui analyse quels composants matériels ou logiciels sont impliqués et dans quel ordre, du traitement de l'impulsion d'entrée d'un capteur vers l'impulsion de sortie souhaitée de l'actionneur concerné. Le temps d'exécution acceptable le plus long de l'ensemble de la chaîne peut être déduit des conditions physiques du capteur et de l'actionneur. Il peut ensuite être décomposé en temps d'exécution maximal autorisé pour chaque composant. Selon le cas, ils durent de 10 µs à 100 ms. Si le temps d'exécution maximal autorisé est dépassé, cela peut nuire à la commodité et provoquer, par exemple, du bruit. Pour les fonctions de sécurité qui nécessitent des capacités en temps réel, cela peut même comporter un risque d'atteinte à la vie ou à l'intégrité corporelle.

Afin de respecter les exigences temps réel résultantes pour les logiciels et de les implémenter, un système d'exploitation Autosar divise les fonctions logicielles pertinentes en sous-étapes, appelées tâches. Les tâches sont activées par des événements externes, c'est-à-dire qu'elles sont marquées comme exécutables, puis traitées en fonction de la priorité. Pour la plupart des applications, une minuterie avec une table d'activation cyclique est utilisée pour activer les tâches, car les fonctions de contrôle pertinentes doivent traiter en continu les nouvelles générations de signaux. Il en résulte un temps de réponse par défaut qui se répète en cycles de quelques dizaines de ms. Plusieurs étapes sont nécessaires pour s'assurer que les exigences temps réel sont satisfaites. D'une part, les mesures appropriées lors de l'implémentation des étapes de traitement du signal doivent bien entendu garantir qu'elles pourront fournir le temps de réponse requis. D'autre part, l'activation des tâches liées au temps dans l'ECU doit permettre d'implémenter les exigences de temps. À ces fins, le comportement d'exécution de toutes les tâches doit être analysé afin de trouver une hiérarchisation des priorités appropriée et un ordre d'activation chronologique qui garantit la rapidité d'exécution.

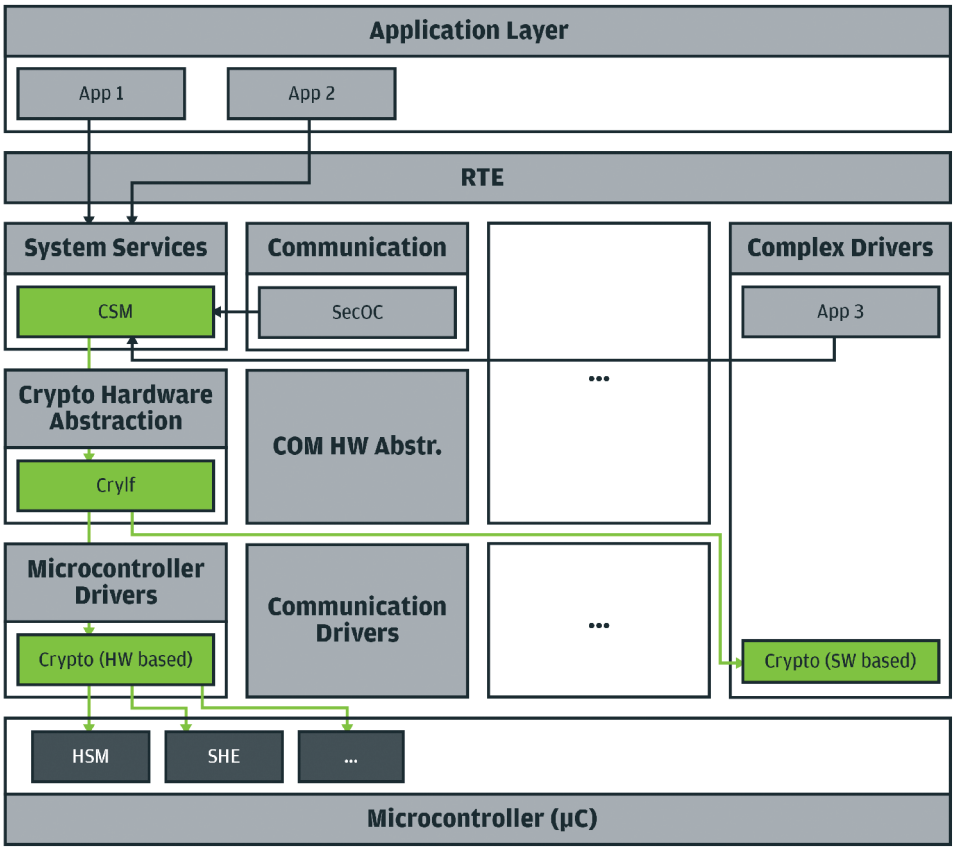
Logiciel de base Autosar

De nos jours, les fonctions d'une ECU ne sont plus seulement des fonctions directement impliquées dans le traitement du signal. Le logi-

ciel de base qui s'occupe, entre autres, de la communication et de la gestion, constitue une partie importante de ces fonctions. Ce logiciel de base doit également être divisé en tâches, analysé en fonction de ses exigences d'exécution et pris en compte dans l'ensemble du système. En résumé, il existe deux ensembles de fonctions : une partie du logiciel de base, par exemple pour communiquer les valeurs de signal, est nécessaire pour permettre le traitement du signal à temps. Cette partie doit bénéficier d'une priorité suffisamment élevée. L'autre partie affiche des exigences en temps réel nettement plus faibles : cette partie doit s'exécuter régulièrement « en arrière-plan » ; cependant, elle peut être brièvement devancée par des fonctions ayant des exigences en temps réel plus strictes. Par conséquent, un modèle d'implémentation commun doit déplacer ces fonctions non critiques vers une tâche de fond de faible priorité. Dans ce contexte, le problème est la hiérarchisation des fonctions entre elles : en général, il y a plusieurs activités d'arrière-plan ; aucune de ces activités ne doit échouer sur une plus longue période. La solution à cette hiérarchisation des priorités est la répartition par permutation circulaire avec laquelle chacune des fonctions de faible priorité calcule pendant une courte période, suivie de la fonction suivante. Dans la terminologie Autosar, chacun des modules impliqués fournit une « Main Function » ou fonction principale avec un temps d'exécution limité. Dans la tâche d'arrière-plan, ces fonctions sont appelées l'une après l'autre.

1 MODULES DE SÉCURITÉ DANS UN SYSTÈME AUTOSAR 4.3

Depuis sa version 4.0, Autosar contient des spécifications pour les logiciels de cryptographie de base. Dans ce contexte, le Crypto Service Manager (CSM) fournit des services cryptographiques à l'application. Par exemple, il est utilisé par le module Secure Onboard Communication (SecOC) qui authentifie par cryptographie les valeurs de signal dans les paquets de données transmis.



Sécurité

Les algorithmes cryptographiques sont un exemple de « fonction d'arrière-plan » classique. Depuis sa version 4.0, Autosar contient des spécifications pour les logiciels de cryptographie de base – les spécifications appropriées ont été révisées et affinées dans les versions ultérieures. Dans ce contexte, le Crypto Service Manager (CSM) fournit des services cryptographiques à l'application (figure 1). Par exemple, il est utilisé par Secure Onboard Communication (SecOC) qui authentifie par cryptographie les valeurs de signal dans les paquets de données transmis. Comme son nom l'indique, le CSM gère les services cryptographiques. Leur implémentation réelle se trouve dans une bibliothèque crypto spécifique au fabricant. Pour prendre en charge l'intégration dans un système Autosar ayant des exigences temps réel, le CSM traite les demandes de manière asynchrone : dans un premier temps, elles sont uniquement sauvegardées puis traitées une par une dans les appels de la MainFunction CSM. A cette fin, la MainFunction CSM, pour sa part, appelle les MainFunctions de toutes les primitives cryptographiques sous-jacentes, chacune d'entre elles calculant ensuite quelques étapes.

Les temps de calcul des primitives cryptographiques typiques dépassent de plusieurs ordres de grandeur ceux des fonctions de traitement du signal. Cela pose un dilemme lors de la division en appels MainFunction : quelle est l'étendue exacte du calcul autorisé ? Trop d'étapes de calcul rendraient la cryptographie inutile en remettant en question les propriétés temps réel de l'ensemble du système. Trop peu d'étapes retarderaient le calcul des opérations cryptographiques plus longues à un point tel que les avantages seraient limités. En outre, un autre effet doit être pris en compte : la MainFunction spécifique est responsable de la gestion de son état de calcul interne de sorte à permettre l'exécution pas à pas du calcul. Cette gestion de l'état, pour sa part, implique un temps système. Moins un appel calcule, plus le temps système est élevé.

C'est au fournisseur de logiciel de trouver une solution standard à ce problème. Dans les scénarios classiques, la cryptographie au moment de l'exécution est, par exemple, utilisée pour authentifier des blocs de données plus petits avec des méthodes de cryptage symétrique.

Une solution peut consister à calculer un bloc symétrique pour chaque appel de la MainFunction. Cependant, avec des méthodes plus complexes, il est difficile de trouver un compromis raisonnable. Il peut s'agir, par exemple, d'authentifier/chiffrer de plus grandes quantités de données ou de générer des signatures asymétriques. Par exemple, supposons qu'il est acceptable de calculer une fois par milliseconde pendant 100 µs – une hypothèse très optimiste qu'il serait difficile de maintenir dans la plupart des scénarios en temps réel. Par rapport au temps de calcul pur, une telle division signifie un décuplement du temps jusqu'à ce qu'un résultat soit disponible. Pour les fonctions cryptographiques dont les temps de calcul purs causent déjà des difficultés, une telle division limite considérablement l'utilisation pratique.

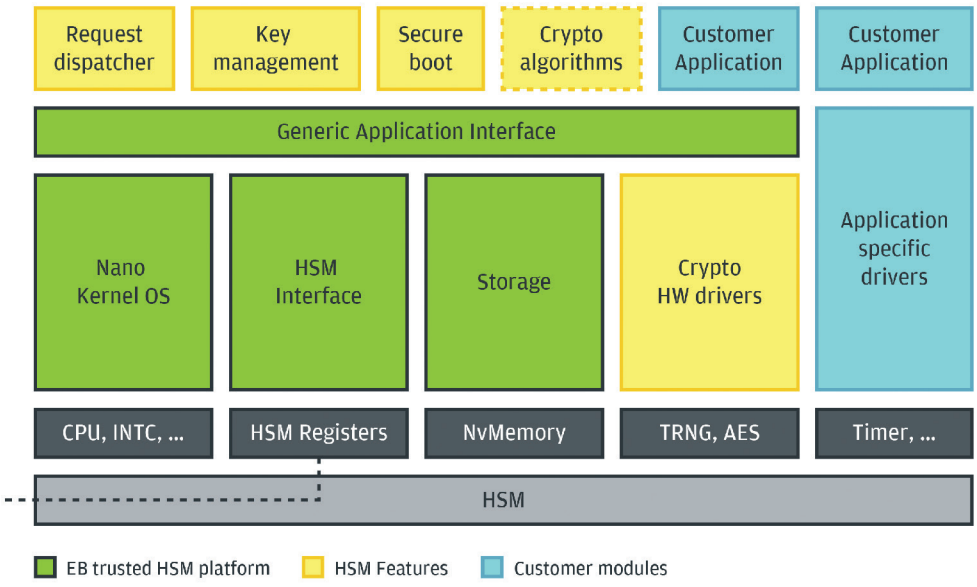
Modules de sécurité matériels HSM

Il est évident que les exigences conflictuelles en termes de capacité en temps réel et de temps système pour les méthodes cryptographiques ne peuvent pas être résolues par le seul logiciel. Par conséquent, une solution évidente consiste à utiliser du matériel spécialisé capable de calculer les algorithmes appropriés – ou une grande partie d'entre eux – en parallèle avec le processeur principal. Ensuite, le CSM Autosar et les bibliothèques cryptographiques associées ne transmettent les requêtes qu'à ce matériel et, dans leur MainFunction, vérifient cycliquement si un résultat est disponible. Le premier de ces coprocesseurs matériels a été désigné « Secure Hardware Extension » (ou extension matérielle sécurisée) par la Hersteller Initiative Software (HIS) déjà au cours de la dernière décennie. En ce qui concerne les algorithmes cryptographiques, cette spécification est encore limitée à l'implémentation de l'AES-128 dans différents modes. Des développements plus récents montrent qu'en raison du grand nombre de cas d'utilisation possibles, un coprocesseur matériel pur est souvent limité et n'est donc pas idéal.

D'où le recours croissant aux modules de sécurité matériels (HSM). Les HSM sont des microcontrôleurs autonomes qui sont reliés aux bus du système hôte par une sorte de pare-feu. Le HSM a normalement sa propre RAM protégée, une zone flash exclusive pour le code programme et les données, et ses propres périphériques tels que des minuteries, des accélérateurs matériels pour certains algorithmes cryptographiques ou des générateurs de nombres aléatoires réels. Il est capable d'accéder au matériel complet de l'hôte. Il est ainsi possible d'implémenter un démarrage sécurisé et authentifié du système ou une surveillance de l'hôte au moment de l'exécution. La mémoire flash de données exclusive peut être utilisée pour stocker des secrets de sorte que le système hôte n'y ait pas accès. Cela signifie que l'hôte peut demander une opération cryptographique effectuée par le HSM sans que la clé ne quitte le HSM. A cet égard, toutefois, l'avantage particulier du HSM est qu'il est librement programmable. En tant que microcontrôleur autonome, le HSM est capable d'exécuter n'importe quel code du programme optimisé pour le cas d'utilisation actuel. Cela lui permet d'implémenter beaucoup plus d'exigences de sécurité qu'un coprocesseur simple.

2 ARCHITECTURE D'UN MICROLOGICIEL HSM OPTIMISÉ

Lorsqu'un système d'exploitation permettant des interruptions est utilisé sur le HSM, le mappage des tâches et la hiérarchisation des priorités peuvent être optimisés en conséquence. De cette façon, les opérations de longue durée peuvent être traitées dans des tâches de faible priorité, tandis que des opérations plus courtes peuvent les interrompre. Avec ce type de mappage, les routines cryptographiques peuvent être implémentées de sorte qu'elles exécutent leur tâche spécifique en une seule session.



Implémentation du micrologiciel HSM

Il peut sembler tentant d'implémenter simplement un logiciel standard Autosar bien établi sur le HSM et d'utiliser des méthodes standard Autosar pour le connecter à l'environnement. Cela permettrait de réutiliser des modèles d'implémentation Autosar familiers. Cependant, ce n'est pas vraiment le cas : les cas d'utilisation d'un système Autosar typique avec traitement du signal en temps réel et un HSM axé sur la sécurité diffèrent considérablement. Cela montre clairement que le micrologiciel HSM peut atteindre une efficacité nettement plus grande s'il est optimisé pour son usage plus librement. De plus, seules des ressources limitées sont à la disposition du matériel HSM actuel – un autre facteur qui rend difficile l'utilisation du logiciel Autosar sur le HSM.

Les cas d'utilisation HSM sont typiquement des modèles client-serveur classiques : l'hôte envoie une ou plusieurs demandes au HSM où elles sont traitées et est informé dès qu'un résultat est disponible. Contrairement à un système Autosar classique, le nombre de tâches de gestion et d'arrière-plan sur un HSM est très limité. On peut donc supposer

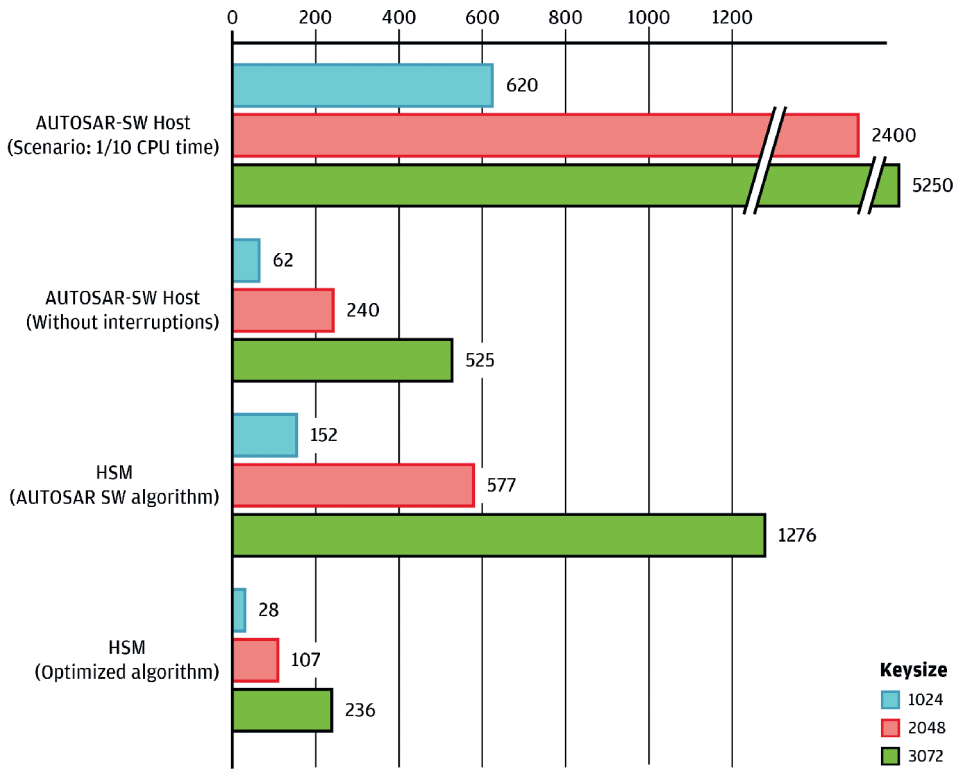
qu'une grande partie du temps de calcul du HSM sera consacrée au traitement des demandes de l'hôte. Lorsqu'un système d'exploitation permettant des interruptions est utilisé sur le HSM, le mappage des tâches et la hiérarchisation des priorités peuvent être optimisés en conséquence. De cette façon, les opérations de longue durée peuvent être traitées dans des tâches de faible priorité, tandis que des opérations plus courtes peuvent les interrompre. Une tâche cyclique hautement prioritaire permet de traiter les tâches de gestion, si celles-ci sont nécessaires. Avec ce type de mappage, les routines cryptographiques peuvent être implémentées de sorte qu'elles exécutent leur tâche spécifique en une seule session. Il n'est plus nécessaire que les interruptions soient autorisées par la routine appropriée elle-même. Elles sont toutefois réalisées – de manière transparente pour la routine – par le système d'exploitation. Le temps système de gestion associé n'est plus requis, ce qui réduit considérablement la taille du code et le temps d'exécution. De plus, les interruptions ne se produisent que lorsqu'elles sont réellement nécessaires – si une seule opération doit être traitée, elle sera calculée sans interruption. La figure 2 montre l'architecture du micrologiciel HSM ainsi obtenu.

Potentiel d'optimisation

Le fait que les performances du matériel HSM sont limitées par rapport au système hôte est un obstacle aux possibilités d'optimisation du logiciel HSM. Pour cette raison, les implémentations logicielles pures sur le HSM devraient théoriquement durer plus longtemps que sur l'hôte. Dans la pratique, cependant, ce sont les économies dues à l'optimisation qui prévalent. La figure 3 montre les mesures pour le RSA sur un système hôte par rapport au HSM associé, nettement plus lent. Ici, l'algorithme logiciel Autosar correspond à l'implémentation du logiciel selon Autosar dans la manière dont il est utilisé sur l'hôte. Pour mesurer le temps de calcul pur, la MainFunction appropriée qui ne se termine pas tant que le calcul n'est pas terminé a été appelée dans une boucle. En conséquence, le temps CPU total est disponible pour le calcul RSA. Les effets

3 TEMPS D'EXÉCUTION DE VÉRIFICATION DE SIGNATURE RSA SUR HÔTE ET HSM

Grâce à son optimisation, le micrologiciel HSM fournit l'ordre de grandeur du résultat de la cryptographie plus rapidement que le processeur hôte de l'ECU, alors que le microcontrôleur HSM associé, est nettement plus lent que ce dernier.



qui résultent du traitement cyclique de la MainFunction et de l'allocation appropriée du temps CPU dans un système réel ont été pris en compte dans un scénario où la MainFunction peut calculer un dixième du temps CPU. Le temps de calcul de l'implémentation Autosar non modifiée et celui d'une implémentation optimisée ont été mesurés sur le HSM. Ce dernier n'a pas de temps système de gestion spécifique à Autosar.

Dans l'exemple de la vérification de signature RSA sur 3 072 bits (exposant présumé : 17), le système hôte nécessite 525 ms de temps de calcul pur. En supposant qu'un dixième du temps CPU soit réellement disponible pour la cryptographie, cela donne plus de 5 s jusqu'à ce qu'un résultat soit disponible. Sur le HSM, le même algorithme a un temps de calcul de 1 276 ms. D'une part, cela montre que la réduction des performances matérielles prolonge significativement les temps de calcul. D'autre part, comme le HSM peut utiliser presque tout le temps CPU pour calculer la fonction appropriée, seul un quart du temps – par rapport au scénario hôte – passe jusqu'à ce que le résultat soit disponible. Ce qui est intéressant, c'est l'algorithme optimisé sur le HSM : il ne nécessite que 236 ms. Grâce à cette optimisation, le micrologiciel HSM fournit ainsi l'ordre de grandeur du résultat plus rapidement que l'hôte – dans un laps de temps correspondant à celui du traitement du signal en temps réel.

Temps pour la cryptographie

L'utilisation d'un HSM avec micrologiciel optimisé débloque donc la possibilité d'une cryptographie à l'épreuve du temps, même pour les ECU temps réel classiques. Même les méthodes de chiffrement asymétrique ne se limitent plus à des modes de fonctionnement spéciaux, mais peuvent être calculées et utilisées au moment d'exécution avec des performances acceptables. Cela rend possible des cas d'utilisation cryptographique qui étaient auparavant impossibles à implémenter. La programmabilité libre du HSM permet des optimisations spécifiques à l'application en termes de sélection des méthodes requises et de leur implémentation. ■