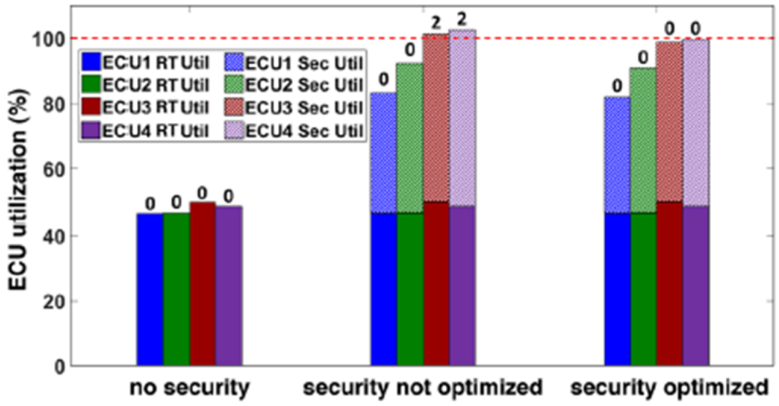
طرح امنیت محور شبکه های حساس به زمان خودرو

خودرو های نوین مجموعه از ECU های به هم متصل هستند که به خاطر افزایش سیستم های مختلف بار پردازشی آن ها دایما رو به افزایش است. این ECU ها لزوما از معماری یا تعداد هسته های پردازشی مشابه برخوردار نیستند. از طرفی نزدیک شدن به پیاده سازی جامع خودرو های خودران باعث افزایش سریع تر تعداد ECU ها شده است. حال که خودرو ها به طور گسترده تری به شبکه های خارج خودرو متصل اند دغدغه امنیت جدی تر شده است. مکانیزم های امنیتی به بخشی جدا نشدنی از ECU ها تبدیل شده اند، اما این مکانیزم ها خود می توانند بار پردازشی اضافی به همراه داشته باشند و منجر به تاخیر در عملکرد بخش های حساس و حیاتی شوند.

از مهمترین تهدید های رایج امنیتی می توان به ماسک، پاسخ و عدم ارایه سرویس اشاره کرد. پروتکل های ارتباطی رایج در خودرو ها از جمله CAN و FlexRay و ... به صورت ذاتی مجهز به ویژگی های امنیتی نیستند که به توانند نگرانی ها را در بحث محرمانگی، اعتبار سنجی و اجازه دسترسی را برطرف کنند. برای بر طرف سازی دسترسی بدون اجازه به طور معمول از روش های کلید های متقارن و غیر متقارن استفاده میشود. هر دو این متد ها می توانند بار پردازشی قابل توجهی را پدید آورند که منجر به تاخیر در اجرای بهنگام شود.



ستون اول استفاده از FlexRay را نشان میدهد بدین صورت که چهار ECU هر کدام دوازده برنامه حساس به زمان را اجرا می کنند. این برنامه بهنگام بوده و ستون دوم و سوم علاوه بر این برنامه ها فعالیت های امنیتی بهنگام لازم را نیز با انجام می رسانند. در ستون دوم بخش فعالیت های امنیتی شامل رمزنگاری و رمزگشایی AES 256 میباشد. در این حالت مجموع بار پردازشی از توان پردازشی فرا تر میرود و ناگزیر منجر به ایجاد تاخیر در پردازش و از عقب افتادن صف رایانش از زمان تعیین شده خود میشود. تلاش نویسنده در این مقاله دست یابی به ستون سوم است که برنامه های بهنگام امنیت و اساسی حساس به زمان ECU ها را بدون سربار پردازشی تاخیر دار، اجرا کند. در این مقاله برای کلید های متقارن استفاده شده است. چرا که بار پردازشی کمتری نسبت به متد نامتقارن دارد.

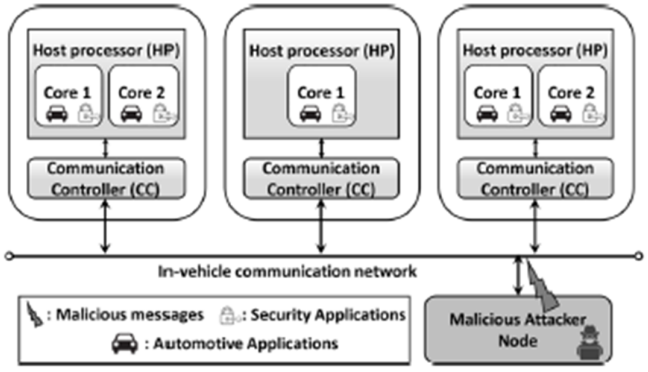
امنیت داده در خودرو ها تا سال های اخیر مورد بی توجهی قرار داشت تا آنکه در سال 2010 گروهی با دسترسی فیزیکی (پورت EBD II) توانستند موفق به تزریق پیام شدند. علاوه بر آن از طریق مهندسی معکوس تعدادی از ECU ها موفق به آپدیت Firmware از طریق شبکه CAN شدند. در سال 2014 عده ای موفق

شدند تا از طریق هک سیستم رادیو خودرو Jeep Cherokee به هر دو شبکه CAN این خودرو دست یابند. در سال های اخیر نویسندگانی موفق به ایجاد برنامه تروجانی شدند که با آلوده کردن گوشی هوشمند پس از متصل شدن آن به سیستم تفریحی/اطلاعاتی خودرو به نویسندگان اجازه می داد تا پیام های دستکاری شده خود را در CAN وارد کنند.

جلوگیری از دسترسی بدون اجازه به گذرگاه داده خودرو های امری سخت می باشد، چرا که هیچ یک از انواع این گذرگاه ها به صورتی ذاتی دارای ویژگی های امنیتی نیستند. برای حل این مشکل استفاده از کد صحت داده ، MACs، پیشنهاد شد. روش های مختلفی برای پیاده سازی این متد با سربار پردازشی کاسته شده در مقالات مختلف ارایه شده است. یک پروتکل صحت داده به نام LCAP معرفی شد که با استفاده از عملگر های Hash ، MACs را Hash میکرد. نویسنده ای دیگر از RC4 برای صحت سنجی بر اساس رمزنگاری معرفی کرد و در مقاله ای دیگر صحت سنجی بر اساس متد سبک وزن PRESENT معرفی شد و بر روی FPGA شبیه سازی شد. اگر چه هر دو این روش های توسط حمله های موفق نویسندگان دیگر کنار گذاشته شدند. اگر چه که اکثر کار های ذکر شده در بالا برای فعالیت های حساس به موقعیت طراحی شده بودند و کاربردی در پردازش های پیچیده تر حساس به زمان ندارند.

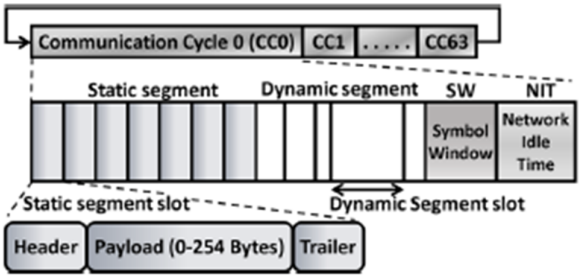
در مقاله ی دیگری متد MACs های رمز شده بر اساس مهر زمان محلیECU و کلید مخفی ارایه شد. اگر چه که این تکنیک نیازمند زمانبندی دقیق بین ECU ها بود و کوچکترین ابهام و اختلافی منجر شکست کلی سیستم میشد. در مقاله ای دیگر به روشی در سطح سخت افزار اشاره شد، در این روش از اینترفیس تقویت شده ای برای شبکه استفاده شده بود که با بهره گیری از ماژول های مبتنی بر سخت افزار به صحت داده ها کمک می کرد. در کار های دیگر از FPGA ها به عنوان پردازنده کمکی برای ECU ها استفاده شد، این FPGA ها امنیت داده ها را بر اساس پروتکل TESLA ارایه میکردند. هر دو روش سخت افزاری بالا به علت نیاز به تغییرات زیاد در ECU ها امکان پیاده سازی به صورت گسترده را نداشتند.

در این مقاله سیستم مورد نظر به شرح زیر می باشد:



سیستمی کلی که در آن ECU ها متعدد با برنامه های مختلف حساس به زمان وجود دارد. این ECU ها به وسیله ی شبکه باس FlexRay به یکدیگر متصل اند. هرECU از دو بخش اساسی HP و CC تشکیل شده است. HP وظیفه پردازش برنامه های خودرو و امنیتی را دارد و CC به عنوان رابط بین HP و FlexRay حضور دارد و در واقع وظیفه ی بسته بندی داده ها و دریافت و فرستادن و فیلتر کردن داده های ناخواسته را دارد. لازم به ذکر است که HP

ها ناهمگون میباشند و از تعداد متفاوتی هسته پردازشی بهره می برند. هر برنامه خودرو دارای پردازش های مستقل و وابسته به هم است. اگر پردازش های وابسته در یک ECU اجرا شوند از حافظه مشترک برای ارتباط استفاده می کنند و اگر در ECU های مجزا اجرا شوند می بایست برای برقراری ارتباط از گذرگاه FlexRay استفاده کنند. برنامه های خودرو از نظر ماهیتی به دو بخش حساس به زمان و حساس به وقایع تقسیم میشوند. برنامه های امنیت محور از جمله برنامه ایربگ ها و ترمز ضد قفل از نوع حساس به زمان هستند و پیغام هایی از این دست را تولید می کنند. برنامه ها حساس به وقایع معمولا توسط برنامه های نگه داری و تشخیص خطا به کار گرفته میشوند. تلاش نویسندگان در این مقاله در ایمن سازی پردازش های حساس به زمان است، چرا که پیغام های ایجاد شده توسط این برنامه ها دارای محدودیت زمانی می باشند. برای این کار از متد های پیشرفته استاندارد رمز نگاری از جمله AES 128,192,256 بیتی و RSA 512,1024,2048, 4096 بیتی و ECC 256,384 بیتی استفاده میشود.



ساختار پروتکل FlexRay به صورت چرخه ای است. بدین معنا که هر بار این چرخه ها با ساختاری یکسان به ترتیب تکرار میشوند. هر چرخه شامل بخش های ثابت ، پویا ، پنجره نماد و زمان بیکاری شبکه است. بخش های ثابت و بیکاری شبکه غیر قابل اجتناب هستند و بخش های پویا و پنجره نماد لزوما همیشه حضور ندارند. بخش ثابت دارای Slot های برابر زمانی است(TDMA)که هر اسلات دارای سه بخش Header, Payload, Trailer است و وظیفه جا به جایی پیغام های حساس به زمان را دارد. بخش پویا وظیفه جا به جایی پیغام های حساس به وقایع را دارد و از بخش هایی با اندازه های مختلف تشکیل شده است. در بخش پویا از Flexible TDMA استفاده شده است. بدین معنا که ECU ای که بالاترین اولویت را دارد بر روی گذرگاه قرار میگیرد. پنجره علامت برای نشانه گذاری اولین دوره ارتباط شبکه است و همچنین برای تعمیر و نگه داری شبکه کاربرد دارد. بخش بیکار شبکه برای حفظ هماهنگی می باشد.

تمرکز این مقاله بر حمله های ماسک و پاسخ است، چرا که این حمله ها مرسوم تر اند و تشخیص آن ها نیز امری مشکل است. حجم افزاینده ی ارتباطات داخلی خودرو های نوین، مسیر های حمله ی فراوانی فراهم می کند. یک هکر می تواند از مسیر های موجود استفاده کند و خود را به عنوان یک ECU در شبکه وارد کند (ماسک) و یا با فرستادن پیغام های معتبر اهداف خود را پیاده کنند. در این مقاله بردار های حمله ای بررسی شده است که مرسوم ترین روش های ارتباط سیستم با محیط بیرونی هستند از جمله سیستم اطلاعاتی/تفریحاتی ، پورت OBD II ، نفوذ به شبکه با استفاده از پراب و جایگزین کردن ECU. بردار های حمله پیچیده تر از جمله دستکاری هسته ها و تروجان ها از موضوعات این مقاله نیستند.

سیستم معرفی شده دارای ورود های زیر میباشد:

تعدادی ECU ناهمگون $N= \{1, 2, ..., N\}$

تعدادی برنامه $A = \{1, 2, ..., \lambda\}$

Task ها $T = \{T1 \cup T2 ... \cup T\lambda\}$ به گونه ای که Ta مربوط به برنامه A می باشد.

هر Task یک ID منحصر به فرد دارد $TID = \{1, 2, ..., G\}$

هر Task به صورت $t_{q,n}$ ارایه میشود بدین صورت که $q \in TID$ و $n \in N$

هر Task توسط چهار فاکتور شناخته میشود $\{\tilde{a}_{q,n}, \tilde{p}_{q,n}, \tilde{d}_{q,n}, \tilde{e}_{q,n}\}$ به طوری که $\tilde{a}_{q,n}, \tilde{p}_{q,n}, \tilde{d}_{q,n}, \tilde{e}_{q,n}$ به ترتیب زمان شروع(رسیدن) ، دوره(مدت)، زمان نهایی(Deadline) و زمان اجرا میباشند.

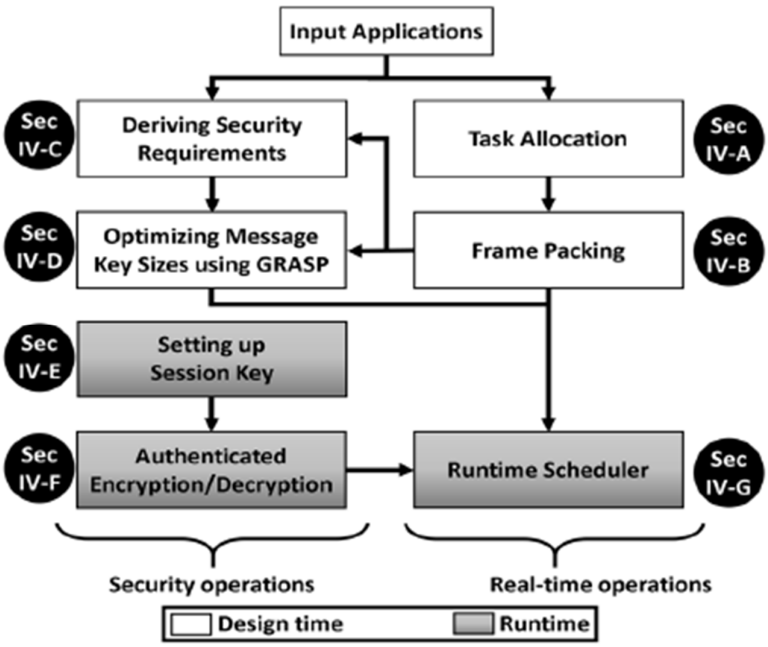
برای هر ECU ، $n \in N$ ، مجموعه سیگنال های جا به جا شده با $Sn = \{S1,n , S2,n ..., Skn,n\}$ نشان داده میشود به طوری که kn تعداد تمام سیگنال ها n است.

هر سیگنال $s_{i,n} \in Sn, (i = 1, 2 ..., Kn)$ ، دارای چهار صفت $\{\bar{a}_{i,n}, \bar{p}_{i,n}, \bar{b}_{i,n}, \bar{d}_{i,n}\}$ می باشد که به ترتیب زمان شروع(رسیدن) ، دوره(مدت)، زمان نهایی(Deadline) و حجم داده به صورت بایت نشان می دهد.

بعد از بسته بندی داده ها در چارچوب هر ECU مجموعه از پیغام ها را دارد

$Mn=\{m_{1,n} , m_{2,n} ,... , m_{Rn,n}\}$ به صورتی که Rn تعداد کل پیغام های n می باشد.

هر پیغام $m_{j,n} \in Mn, (j = 1, 2, ..., Rn)$، دارای پنج ویژگی $\{a_{j,n}, p_{j,n}, d_{j,n}, b_{j,n}, \Delta_{j,n}, \psi_{j,n}\}$ است که به ترتیب زمان شروع(رسیدن) ،دوره(مدت)، زمان نهایی(Deadline) ، حجم داده و کمترین نیاز امنیتی را شامل میشود.



در بخش اختصاص Task ها ، SEDAN به سرعت هر Task را به یک ECU ارایه می دهد تا بتواند حالت بهنگام بودن ECU را حفظ کند. در این بخش Task هایی که به هر دلیلی به یک ECU خاص تعلق دارند از این مستثنی هستند. برای هر Task مصرف بلادرنگ از نسبت زمان اجرا بر دوره

Task بدست می آید. مصرف بلادرنگ برای هر ECU از مجموع مصرف بلادرنگ Task هایی که به آن ECU مشخص تخصیص داده شده است بدست می آید.

$$\tilde{U}_{t_q} = \frac{\tilde{e}_q}{\tilde{p}_q}$$
$$\tilde{U}_n = \sum_{q=1}^{G_n} \left(\tilde{U}_{t_{q,n}}\right)$$

تلاش لازم در این بخش متمرکز بر به صفر رساندن مصرف بلادرنگ ECU ها میباشد.

فرایند تخصیص Task ها به صورت زیر انجام می شود.:

ابتدا لیستی از ECU ها بر اساس مصرف بلادرنگ ECU ها ایجاد میشود، سپس اولین Task تخصیص نیافته به ECU ای که کمترین مصرف بلادرنگ را دارد تخصیص داده میشود و پس از آن مصرف بلادرنگ آن Task به مصرف بلادرنگ ECU اضافه میشود و این رویه برای Task بعدی تکرار میشود تا هیچ Task اختصاص نیافته ای باقی نماند.

$$m_{j,n}^{SV} = m_{j,n}^{AW} * m_{j.n}^{SS}$$
$$Aggregate\ Security\ Value\ (ASV) = \frac{\sum_{n=1}^{N}\sum_{j=1}^{R_n}(\psi_{j,n} * m_{j,n}^{SV})}{\sum_{n=1}^{N} R_n}$$

ASV معیاری بسیار مهمی برای تعیین امنیت به ازای تعداد پیغام ها است. از این معیار می توان برای مقایسه امنیت در سیستم های مختلف استفاده کرد. تعریف معادله پایین از ضرب نمره امنیتی در وزن ASIL آن بدست می آید، که به صورت زیر نشان داده میشود و به آن ارزش $m_{j,n}^{SV} = m_{j,n}^{AW} * m_{j.n}^{SS}$ امنیتی می گویند.

نرخ میزان شکست بنابر ASIL را با خطا در زمان (FIT) نشان می دهند. FIT را به صورت ماکزیمم خطا های قابل پذیرش در 1 میلیارد ساعت استفاده بیان می کنند. بنابر قوانین ASIL نوع A،B،C،D به ترتیب 1000FIT،100FIT،100FIT،10FIT می باشد. بنابر ASILD تعداد خطا ها در یک ساعت باید کمتر از 10^-8 باشد. بنابر K کلمه موجود در MAC احتمال خطای آن 2 به قوه $((3600*10^3)/p_{j,n})*2^{-k} \leq 10^{-8}$ K می باشد و Pj,n دوره یک پیغام را در میلی ثانیه نشان می دهد. بنابر این مینیمم تعداد بیت MAC به صورت زیر به دست می آید.

$$\Delta_{j,n}(D) = k \geq \left\lceil Q + \log_2\left(\frac{1}{p_{j,n}}\right)\right\rceil$$

ثابت Q برای ASIL D مقداری برابر 48,35 دارد. مقدار Q برای درجه بندی های C و B و A دارای ارزش 45,04 و45,04 و 41,72 .

مصرف ECU برای انجام رمز نگاری بلوکی برای یک پیام به صورت زیر محاسبه میشود:

$$\overline{U}_{m_{j,n}} = \left(\left\lceil \frac{b_j}{b_{size}} \right\rceil * \frac{T_{encr/decr}}{p_j} \right)$$

در فرمول بالا منظور از b size اندازه بلوکی است که بر روی آن امر رمزنگاری و یا رمزگشایی انجام میشود.

$$\overline{U}_{m_{j,n}} = \left(\left\lceil \frac{b_j}{16} \right\rceil * \frac{T_{AES(X)}}{p_j} \right)$$

از آن جایی که از AES استفاده میشود می توان معادله اول را به صورت بالایی بازنویسی کرد. بنابر این برای محاسبه مصرف ایجاد شده توسط بخش امنیت، مصرف ایجاد شده توسط بخش امنیت از تمامی داده های فرستاده شده و دریافت شده را جمع میشود. برای محاسبه مصرف کل یک ECU از جمع مصرف ایجاد شده توسط امنیت و مصرف بلادرنگ استفاده میشود. برای جلوگیری از تاخیر در اجرای هر ECU باید از مصرف کل آن کمتر از 100% باشد.

$$\overline{U}_n = \sum_{j=1}^{R_n} \overline{U}_{m_{j,n}}$$
$$U_n = \widetilde{U}_n + \overline{U}_n$$

پس از تعیین طول کلید مورد نیاز بر اساس طول داده های بسته بندی شده و حدود امنیت بدست آمده برای آن، عملی بودن ان چک میشود، بدین معنا که مجموع مصرف ECU و حدود های زمانی از دست رفته را بررسی می کنند و در صورتی که امکان آن وجود داشته باشد به متد متاهیوریستیکی GRASP ارایه میشود. و نتیجه آن در نهایت باعث تصمیم گیری صحیح و ایجاد جدول زمانی مناسب میشود.

Framework	Lin et al. 128	Lin et al. 192	Lin et al. 256	SEDAN
Low load	28	12	0	0
Medium load	45	16	0	0
High load	96	31	0	0

در نهایت در جدول بالا مقایسه تعداد اشتباهات ایجاد شده در روش Lin et al که یک روش رقیب برای SEDAN است را نشان میدهد. این تعداد دفعات عدم رعایت امور امنیتی برای بار های پردازشی مختلف را نشان میدهد. تنها Lin et al. با مدل AES 256 بیتی می تواند رقیبی برای SEDAN به حساب آید اما Lin et al. بر خلاف SEDAN از روش هوشمند اختصاص کلید بهره نمی برد و همچنین SEDAN از ارایه کلید بر بستر گذرگاه نا امن خودداری می کند. این درحالی است که علاوه بر نکات گفته شده SEDAN برای یک Session نیاز به تعدادی جا به جایی کلید ندارد و این موضوع در Lin et al. اجباری است. تصویر پایین مقایسه ای بر اساس ASV می باشد که در آن میزان بار پردازش متفاوت نمایش داده است. مشخصا Lin et al.265 ایمن ترین روش است ولی در طول اجرای آن چند Deadline عقب افتاده وجود دارد و سربار پردازشی آن بالاتر از SEDAN است. SEDAN با ارایه مینمم اصول امنیتی از سر بار پردازشی و عقب افتادن زمان های مقرر جلوگیری می کند.

از ایرادات شاخص این مقاله یکسان پنداری روش جا به جایی کلید است. اگر چه در این روش برای ارایه کلید از متد های ریاضیاتی غیر متصل استفاده میشود ولی همچنان روش Lin et al. از نظر تکرار درخواست کلید

برای یک Session ایمن تر است. همان طور که در این مقاله هم اشاره شده است گسترش کامپیوتر های کوانتومی چالشی برای متد ها گذشته است پس بهتر بود برای رمز نگاری ، نویسنده قدری آزادی عمل ایجاد می کرد.