Gao et al. Cybersecurity (2021) 4:3
https://doi.org/10.1186/s42400-020-00066-2

Cybersecurity

RESEARCH Open Access

A lightweight cryptographic algorithm for the transmission of images from road environments in self-driving

Runchen Gao[1,2*], Shen Li[1], Yuqi Gao[2] and Rui Guo[1]

Abstract

With the large-scale application of 5G in industrial production, the Internet of Things has become an important technology for various industries to achieve efficiency improvement and digital transformation with the help of the mobile edge computing. In the modern industry, the user often stores data collected by IoT devices in the cloud, but the data at the edge of the network involves a large of the sensitive information, which increases the risk of privacy leakage. In order to address these two challenges, we propose a security strategy in the edge computing. Our security strategy combines the Feistel architecture and short comparable encryption based on sliding window (SCESW). Compared to existing security strategies, our proposed security strategy guarantees its security while significantly reducing the computational overhead. And our GRC algorithm can be successfully deployed on a hardware platform.

Keywords: 5G, Internet of things (IoT), Mobile edge computing, Feistel architecture, SCESW, GRC algorithm

Introduction

As industry continues to grow, the conditions required for production are becoming more and more complex, the most important of which are stability, efficiency and high concurrency. Currently, 5G networks are mainly serving industrial production, and 5G networks meet the needs of ultra-stable and large-scale machine connectivity in industrial production by supporting three business scenarios: Enhanced mobile bandwidth (eMBB), Ultra-reliable low latency (uRLLC), and Massive Machine Type Communication (mMTC). Mobile edge computing provides local big data triage, flexible routing scheduling, efficient cloud computing and cloud mega-storage capabilities, which makes it be an essential part of the Industrial Internet of Things.

The reason we use mobile edge computing is because all three scenarios in a 5G network are inextricably linked to it. The eMBB required for high bandwidth and high concurrency is due to the large number of endpoints that impose a greater data traffic impact on the core network (i.e., 5G Core), with the gateway responsible for data forwarding at each endpoint becoming a bottleneck for the entire network. The local big data triage, flexible routing, and other features provided by mobile edge computing can effectively relieve data transmission pressure on the core network. uRLLC with low latency limit requirements that impose stringent requirements on the network. Technologies, such as local service processing, content acceleration and so on, provided by mobile edge computing significantly reduce the transmission time of data streams in the core network. mMTC is provided many resource-constrained IoT terminals that cannot achieve high power consumption for computing, storage, etc. Mobile edge computing provides computation, storage capabilities close to IoT terminals.

Problem of image transmission in self-driving as an example to proposed encryption algorithm for self-

* Correspondence: runchengao@stu.xupt.edu.cn
[1]School of Cyberspace Security, Xi'an University of Posts &Telecommunications, Xi'an 710121, China
[2]School of Telecommunication and Information Engineering, Xi'an University of Posts & Telecommunications, Xi'an 710121, China

driving image transmission. Fourthly, we analyze the security performance in GRC algorithm. Lastly, we practical GRC algorithm verification through self-designed hardware platform.

Various security issues facing mobile edge computing

Potential security vulnerabilities in edge computing environments have an edge access level, an edge management level, and we categorize these vulnerabilities into multiple scenarios that can appear in the following

A. Insecure communication protocols between terminals and remote ends: As most of the edge nodes and massive, heterogeneous, resource-constrained mobile devices use short-range wireless communication technologies, and most of the edge nodes and cloud servers use message middleware or network virtualization technologies, most of these protocols have insufficient security considerations. For example, in industrial edge computing, enterprise and IoT edge computing scenarios, there are numerous insecure communication protocols between sensors and edge nodes (e.g., Wi-Fi, GSM, etc.), lack of encryption, authentication and other measures, and easy to be eavesdropped and tampered with; in telecom operator edge computing scenarios, the wireless communication protocol between edge nodes and users is based on WPA2, and the message middleware between cloud servers and edge nodes is based on instant messaging protocol, and network construction and expansion of network devices at the edge is carried out through the network Overlay control protocol, the main consideration is communication performance, with insufficient consideration of the confidentiality, integrity, authenticity and undeniability of messages.
B. Data located at edge nodes is highly vulnerable to destruction: the lack of effective data backup, recovery, and auditing measures due to the infrastructure of edge computing located at the edge of the network leads to the possibility that attackers may modify or delete user data on edge nodes to destroy some evidence. In the enterprise and IoT edge computing scenarios, using the traffic regulation scenario as an example, the edge node at the side of the road holds video of a traffic accident reported by a nearby vehicle, which is important evidence for accident forensics. Criminals may attack edge nodes to forge evidence to get out of punishment. Furthermore, in a telecom operator edge computing scenario, in the event that subscriber data is lost or corrupted on the edge node or server and there is no backup of the corresponding subscriber data in the cloud and no effective mechanism is provided by the edge node to recover the data, the subscriber is forced to accept such loss; if the above scenario occurs in an industrial edge computing scenario, the loss or corruption of data on the edge node will directly affect the batch industrial production and decision-making process.
C. Account information is vulnerable to hijacking: account hijacking is a kind of identity theft, the main target is generally the field equipment users, attackers in a dishonest way to obtain equipment or services tied to the user-specific unique identification. Account hijacking is usually done through phishing emails, malicious pop-ups, etc. In this way, users often inadvertently leak their own authentication information. This is used by attackers to perform malicious operations such as modifying accounts and creating new accounts. In industrial edge computing, enterprise and IoT edge computing scenarios, users' field devices are often directly connected to fixed edge nodes, and the devices' accounts often use weak, easy-to-guess and hard-coded passwords, making it easier for attackers to disguise themselves as legitimate edge nodes to conduct phishing, spoofing and other operations on users. In a telecom operator edge computing scenario, where a subscriber's end device often needs to move between edge nodes and switch access frequently, an attacker can easily intercept or illegally obtain account information that the subscriber has authenticated to use by compromising an edge node that the subscriber has already passed through, or forging a legitimate edge node.
D. False malicious edge nodes: In the edge computing scenario, the number and type of entities involved is large and the trust situation is complex. Attackers may disguise malicious edge nodes as legitimate edge nodes and trick end users into connecting to malicious edge nodes to covertly collect user data. In addition, edge nodes are often placed near users, at locations such as base stations or routers, and even at the extreme network edges of wireless access points, making it very difficult to provide security for them, and physical attacks are more likely to occur. Existing intrusion detection techniques are difficult to detect the above attacks due to differences in edge computing device architecture, protocols, and service providers.
E. Insecure Application Program Interface (API): In a cloud service environment, to facilitate user interaction with cloud servers, a series of user interfaces or API programming interfaces are

opened that need to prevent accidental or malicious access. In addition, third parties often develop more value-added services based on these interfaces or APIs, which introduces a new layer of more complex APIs, with a corresponding increase in risk. Therefore, whether in the industrial edge computing, enterprise and IoT edge computing scenarios, or carrier edge computing scenarios, there is a need to pay attention to interface security.

F. Advanced Persistent Threat Attack (APT): An APT attack is a parasitic form of attack that typically establishes a foothold in the target infrastructure from which data is surreptitiously stolen and security measures can be adapted to protect against APT attacks. In an edge computing scenario, APT attackers first look for vulnerable edge nodes and try to attack them and hide themselves. To make matters worse, edge nodes often have many known and unknown vulnerabilities and suffer from untimely synchronization with security updates in the central cloud. Once breached, coupled with the current edge computing environment's inadequate ability to detect APT attacks, the user data and programs connected to that edge node are insecure. What is more threatening than traditional network APT is that in industrial edge computing, enterprise and IoT edge computing scenarios, the default settings of field devices and networks are simple and mostly insecure, and edge centers cannot provide effective mechanisms to modify these configurations in a timely manner, making APT attacks more susceptible and propagating, easily spreading to a large number of field devices and other edge nodes.

G. Less secure end-user privacy data: Edge computing migrates computing from the cloud to the nearest end-user and directly processes and makes decisions about the data locally, avoiding to some extent the spread of data over long distances in the network and reducing the risk of privacy leakage. However, because edge devices acquire first-hand data from users, they have access to a large amount of sensitive and private data. For example, in a telecom operator edge computing scenario, it is extremely easy for curious users of edge nodes to collect and snoop on other users' location information, service content, frequency of usage, etc. In industrial edge computing, enterprise and IoT edge computing scenarios, edge nodes lacking effective encryption or desensitization measures relative to traditional cloud centers, and any information they store will be compromised in the event of a hacking attack.

Edge computing security issues in self-driving

Self-driving is accomplished by the collaboration of its supporting software and hardware, which mainly includes heterogeneous hardware platform, system software and application software. The heterogeneous hardware platform consists of a computing unit, an AI unit and a control unit. The computing unit is usually a multi-core ARM chip that runs system software and applications related to self-driving. The AI unit uses GPU, AI chip, FPGA and other parallel computing architecture chips, and relies on system software for resource allocation and scheduling to complete the AI processing of image and LIDAR data. The flow and distribution relationship of data from each terminal in self-driving is shown in Fig. 1.

Safety issues and needs for automated driving

From the traditional car closed scenario to the open scenario of automatic driving, automatic driving software logic is complex, the amount of code, security vulnerabilities are difficult to avoid, once attacked, is likely to cause car damage or even more serious public safety problems. It also faces a broad, new type of attack window, which mainly includes.

(a) Physical attack window: contact-based attacks via physical interfaces such as USB and Ethernet.
(b) Proximity attack window: non-contact attacks through proximity communications such as Wi-Fi and Bluetooth.
(c) Remote attack window: non-contact attacks via 5G, GNSS, etc. communication or navigation.

Data acquisition on road conditions in self-driving

In automated driving, the data of the road ahead is mainly acquired through LIDAR point cloud data or through millimeter wave radar. Echoes collected data, but after both types of data acquisition, the first thing that happens is that the chip converts the collected data into images, into After completion, it enters the AI unit for target recognition and decision making by the upper control system. If an intruder destroys the picture data during this process, it can directly threaten the safety of the traffic, and for such a phenomenon, we propose a image encryption algorithm in an self-driving scenario to deal with the above problem. Security and comparison complexity of this scheme in the phase of encryption is $O(log(m_2)log^3 p)$.

GRC algorithm design

The GRC algorithm proposed in this paper is for simple structural algorithms that can be implemented in low-power devices in an IoT environment. Popular ciphers include a series of SF (Ebrahim and Chong 2013) and

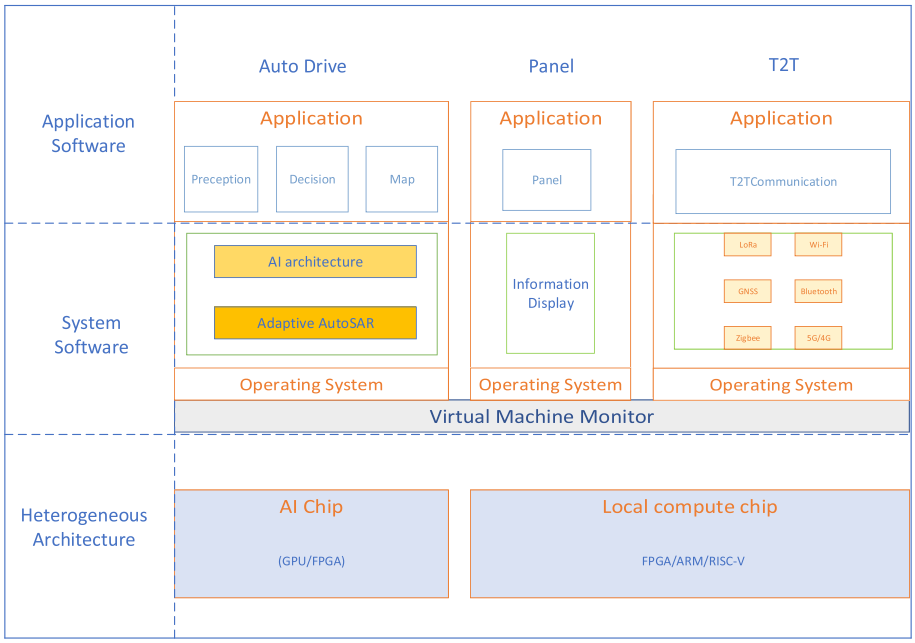

Fig. 1 Software & Hardware in self-driving

DES (Coppersmith 1994) ciphers, and they all have in common the use of the feistel architecture. One of the main advantages of using the feistel architecture is that the encryption and decryption operations are almost identical. The algorithm proposed in this paper is a hybrid approach based on feistel and SCESW. SCESW is a combination of short comparable encryption (SCE) The scheme is combined with a new algorithm with windowing technology. Since the original comparative encryption scheme has a large storage and computational overhead, the new scheme can make the storage and computation of the encryption algorithm securely. Overhead and computational overhead are reduced, resulting in increased efficiency. The logarithm of the sliding window method used in this paper no longer distinguishes between zero and non-zero windows, but instead opens them uniformly, so that each window size equivalence. The disadvantage of symmetric cryptography is that both parties to the transaction use the same key, which does not guarantee security. In addition, each time a pair of users uses the symmetric encryption algorithm, they need to use a unique key that is not known to the others, which makes it difficult to send and receive messages. The number of keys owned by both sides has increased geometrically and key management has become a burden on users. Symmetric cryptographic algorithms are more difficult to use on distributed network systems, mainly because of the difficulty of key management and the high cost of use. In contrast to public key encryption algorithms, symmetric encryption algorithms provide encryption and authentication but lacking signature capabilities, making the use of the range is reduced. In symmetric key algorithms, the encryption process consists of cryptographic rounds, each of which is based on some mathematical function to generate obfuscation and diffusion. The increase in the number of rounds ensures better security, but ultimately leads to an increase in the consumption of bound energy (Chandramouli et al. 2006). The algorithm utilizes a feistel network of alternative diffusion functions to drastically reduce the number of cryptographic rounds of the system. Therefore, the GRC algorithm is designed using the characteristics of the above two methods. It presents substantial security in the IoT environment while keeping the computational complexity at a moderate level.

The GRC scheme consists of five algorithms: parameter generation G, data chunking P, label generation D and Algorithm E for secret message generation and Algorithm C for secret message comparison. (Meng et al. 2018)

Before specifying the GRC scheme, a definition of the symbols used in the GRC scheme is given in Table 1.

Algorithm G

Given safety parameters $k \in N$ and range parameter $n \in N$. Algorithm output public parameter *para* and main key *key* as shown in Eq. (1).

$$G(k, n) = (para, key) \tag{1}$$

The *para* in Eq. (1) is $para = (n_1, H_1, H_2, H_3)$ and select H_1, H_2, H_3 satisfies the condition $\{0,1\}^h \times \{0,1\}^* \rightarrow \{0,$

Table 1 Definition of the symbols used in the GRC scheme

Symbol	Definition
H_1	$Hash_1$ Function
H_2	$Hash_2$ Function
H_3	$Hash_3$ Function
H	Hash Function
key	Main Key
l	k bits Random Number
n	numerical length
h	Number of Hash Functions
m	Number of windows
$para$	Output Parameters
$\oplus$	XOR
$\odot$	XNOR
$\|\|$	Connections

$1\}^k$. In this paper, we take a 64-bit block cipher example. After specific operations are performed on a specific key given by the user to cause obfuscation and proliferation, the specific key will generate five unique keys. And encryption or decryption shall be used these keys. The process of the *key* is shown in Fig. 2.

Next, we explain the individual steps in Fig. 2

- First, we divided the 64-bit user-given main key (*key*) equally into multiple 4-bit parts.
- Second, we obtain the *f*-function we need by initial substitution of 16-bit of data made up of four 4-bit data at the same location as the previous quadratic part. The results obtained are shown in Eq. (2).

$$key(i)a = \|_{j=1}^{4} key_{4(j-1)+i} \tag{2}$$

- Third, we get $key(i)b$ by every 16-bit data segments transform from *f*-function. The results obtained are shown in Eq. (3).

$$key(i)b = f(key(i)a) \tag{3}$$

- Together, the P and Q tables form the f-function. It is because these tables perform linear and nonlinear transformations that the obfuscation and diffusion processes in symmetric encryption are completed, as shown in Fig. 3. And P and Q tables are shown in Tables 2 and 3.
- Fourth, the key $key_i(i = 1, 2, 3, 4)$ output from last step as four 4 × 4 matrix are shown in Eqs. (4)(5)(6)(7) below:

$$key1 = key1a(j) \tag{4}$$

$$key2 = key2b(j) \tag{5}$$

$$key3 = key3c(j) \tag{6}$$

$$key4 = key4d(j) \tag{7}$$

where j = 1 to 16.

- Fifth, we want to obtain round key in symmetric encryption, last step four matrix will transformed to 16 bits data are called $k_i(i = 1, 2, 3, 4)$ shown in Eqs. (8)(9)(10)(11) below:

$$\begin{aligned} k_1 = {} & a_4\#a_3\#a_2\#a_1\#a_5\#a_6\#a_7\#a_8\#a_9 \\ & \#a_{10}\#a_{11}\#a_{12}\#a_{13}\#a_{14}\#a_{15}\#a_{16} \end{aligned} \tag{8}$$

$$\begin{aligned} k_2 = {} & b_1\#b_5\#b_9\#b_{13}\#b_{14}\#b_{10}\#b_6\#b_2\#b_3 \\ & \#b_7\#b_{11}\#b_{15}\#b_{16}\#b_{12}\#b_8\#b_4 \end{aligned} \tag{9}$$

$$\begin{aligned} k_3 = {} & c_1\#c_2\#c_3\#c_4\#c_8\#c_7\#c_6\#c_5\#c_9\#c_{10} \\ & \#c_{11}\#c_{12}\#c_{16}\#c_{15}\#c_{14}\#c_{13} \end{aligned} \tag{10}$$

$$\begin{aligned} k_4 = {} & d_{13}\#d_9\#d_5\#d_1\#d_2\#d_6\#d_{10}\#d_{14}\#d_{15} \\ & \#d_{11}\#d_7\#d_3\#d_4\#d_8\#d_{12}\#d_{16} \end{aligned} \tag{11}$$

- End, we obtain final round key $k5$ by XOR operation $k1$ to $k4$ as shown in Eq. (12).

$$k5 = k1 \oplus k2 \oplus k3 \oplus k4 \tag{12}$$

Algorithm P

$num = (b_0, b_1, \ldots b_n - 1)$; $b_i \in \{0, 1\}$ is a binary representation of the given number, the output value after counting the open window is $num = (B_0 \ldots, B_{m-1})$; $\frac{n}{m} = t$.

Algorithm D

Given the public parameters *para*, the main key *key* and the number *num*, the algorithm outputs the label *token* as shown in Eq. (13).

$$token = D(para, key, num) \tag{13}$$

The label *token* consists of d_i like $token = (d_1, d_2, \ldots, d_m)$,the d_i is as shown in Eq. (14)

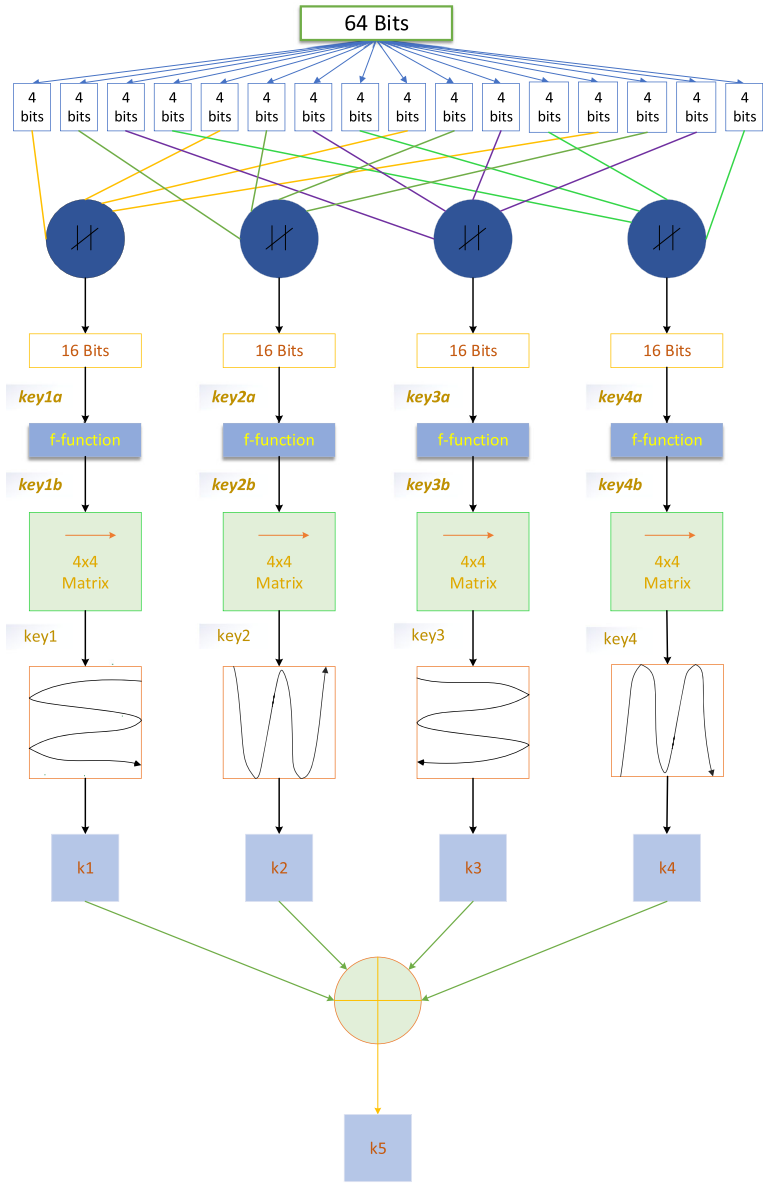

Fig. 2 Process of the key

$$d_i = H_1(key, B_m, B_{m-1}, \ldots, B_i), i = 1, 2, \ldots, m\mathrm{d_i} = \mathrm{H_1(key, B_m, B_{m-1}, \ldots, B_i), i = 1, 2, \ldots, m} \quad (14)$$

Algorithm E

Given the public parameter *para*, the master key *key* and the number *num*. Algorithm E is random to generate $I \in \{0,1\}^k$ and label $ken = (d_1, \ldots, d_m)$, generate f_i according to Eq. (15), output a secret message $ciph = (I, (f_0, f_1, \ldots, f_{m-1}))$. In order to make the secret message length shorter, $(f_0, f_1, \ldots, f_{m-1})$ convert to integer $F = \sum_{i=0}^{m-1} f_i \left(2^{(t+1)} - 1\right)^i$ to save.

$$f_i = H_1(d_{i+1}, I) + H_2(key, d_i + 1) + B_i Mod\left(2^{(t+1)}\right) \quad (15)$$

Then, to create confusion and diffusion during symmetric encryption, this process consists of a number of logical operations, left shifts, swaps and substitutions, and the process shown in Fig. 4.

The $Ro_{i,j}$ in Fig. 4 is

$$Ro_{i,j} = \begin{cases} P_{x_{i,j}} \odot K_i; j = 1 \text{ and } 4 \\ P_{x_{i,j+1}} \oplus Ef_{l_i}; \quad j = 2 \\ P_{x_{i,j-1}} \oplus Ef_{r_i}; j = 3 \end{cases}$$

The final cipher text(*ciph*) is shown in Eq. (16).

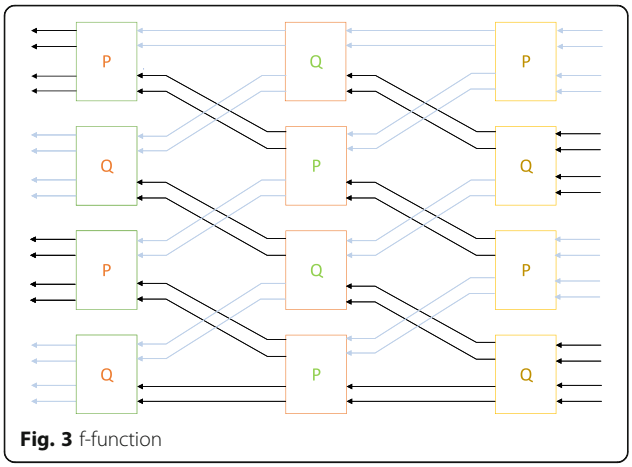

Fig. 3 f-function

$$ciph = R_{51}\#R_{52}\#R_{53}\#R_{54} \tag{16}$$

Algorithm C

Given the public parameter param, the secret messages *ciph*, *ciph** and the label corresponding to one of the secret messages token the output of the algorithm as shown in Eq. (17).

$$\mathrm{Cmp} = \begin{cases} -1, num > num^* \\ 0, num = num^* \\ 1, nnm < num^* \end{cases} \tag{17}$$

The parameter generation algorithm is used to generate the public parameters *para* and master key *key* used in the next steps. The label generation algorithm is used to generate the label *token* associated with the number m^* . The *token** is similar to this process. The secret message generation algorithm is mainly used to generate the secret message *ciph* associated with the number *num*, where the number *num** is generated by the The ciphertext *ciph** is similar to this process. The cryptographic comparison algorithm mainly uses the previously generated cryptographic data and the label associated with one of the numbers to perform the comparison. Finally, we determine the difference between the first different windows of the pair of secret *ciph* and $ciph^*$.

Security analysis

The plaintext that the user needs to transmit is encrypted with a key to obtain a secret message that can be transmitted securely. An attacker intercepts the secret message on the data link and attempts to recover the plaintext by cracking the key. If an adversary is able to decrypt the key, the message is considered to be cracked. If the attacker is sometimes able to decrypt the secret message but is unsure of the key content, the secret message is said to be partially cracked. We assume that the attacker is able to fully intercept the encrypted secret message transmitted over the data link, and that the attacker may also have some additional information, but to assess the security of the secret message, the attacker's ability to crack the computation must also be taken into account. Since the GRC algorithm is an integrated algorithm that combines the feistel architecture and the SCESW scheme, it can be used by Conclusions are drawn from the previously existing security analyses. In the following, the existing security analyses for both components are reviewed and their relevance to the proposed algorithm is discussed.

Table 2 P Table

key_i	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$P(key_i)$	7	5	A	F	D	1	9	0	C	6	2	E	8	4	B	3

Table 3 Q Table

key_i	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$Q(key_i)$	5	8	A	7	C	0	B	3	1	E	2	6	4	F	9	D

Weak Indistinguishability

The SCESW scheme satisfies weak indistinguishability if the H_1, H_2, H_3 functions are pseudo-random functions (Meng et al. 2018).

It is assumed that there is a polynomial time. After adversary Alice inquires, $\mathrm{Adv}^k_{C,A} \coloneqq |\Pr(Exp^k_{C,A} = 0) - \Pr(Exp^k_{C,A} = 1)|$ is not negligible for k in the weakly distinguishable game. Therefore, it can be expressed as $|\Pr(Exp^k_{C,A} = 0) - \Pr(Exp^k_{C,A} = 1)| \geq \varepsilon$, and H_1, H_2, H_3 can be distinguished from the random function. This contradicts the premise of this article and assumes that H_1, H_2, H_3 are pseudo-random functions.

Interpolation attacks

The interpolation attack relies on the simple structure of the cipher component, which may yield a reasonable expression with convenient complexity. The S-box and diffusion layer expression of the proposed algorithm makes this type of attack infeasible. (Muhammad and Irfan 2017)

Weak keys

Nonlinear operations depend on the actual key value of the cipher to map out block ciphers with detectable weaknesses. This occurs in (Daemen 1995). However, the proposed algorithm does not use the actual key in the cipher, but rather XORed it before feeding it to the *f*-function. In the *f*-function, all the non-linearity is fixed and there is no restriction on the selection of the key.

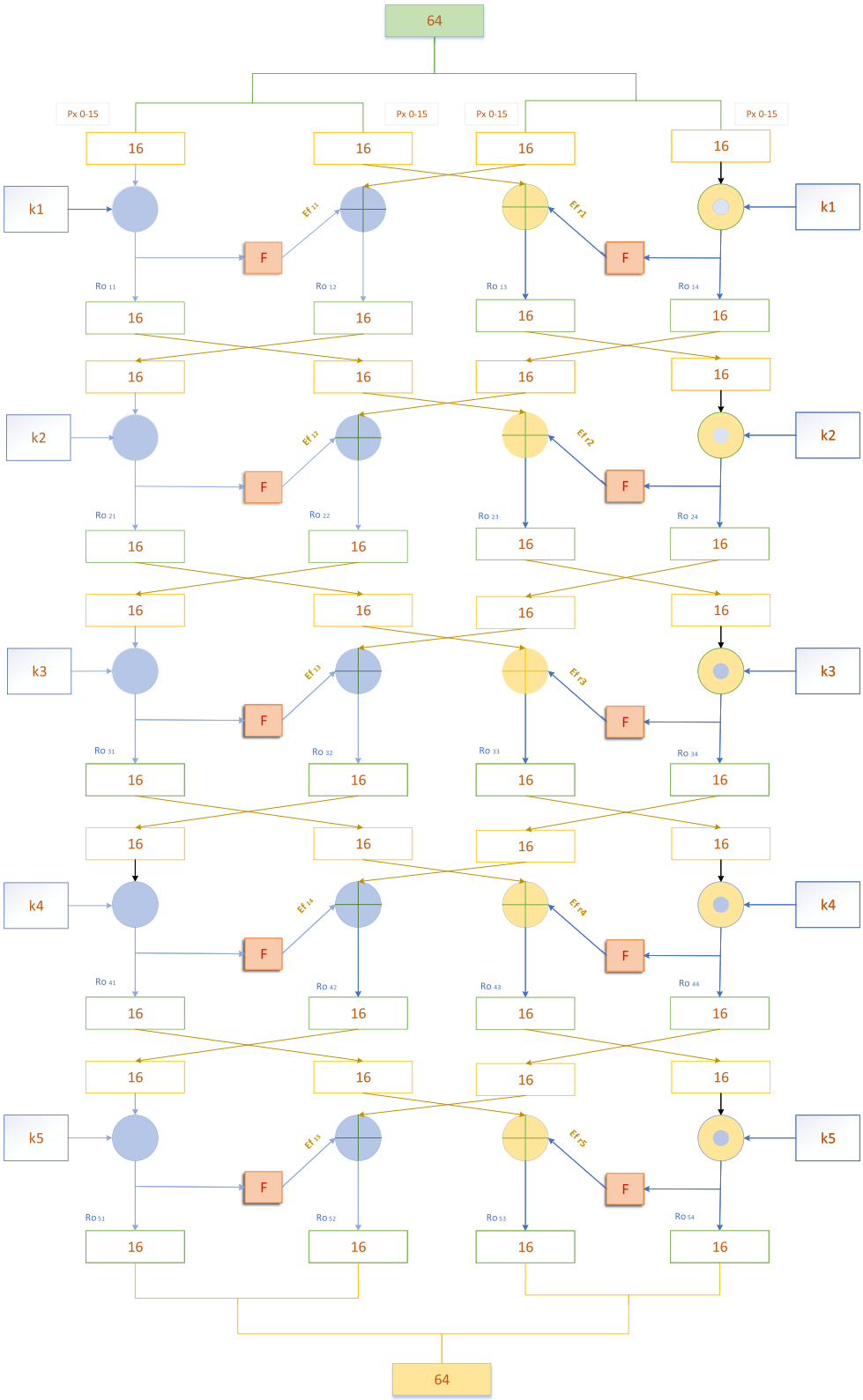

Fig. 4 Encryption Process

Table 4 Different Algorithm Implementations

CIPHER	DEVICE	Block Size	Key Size	Code Size	RAM (Byte)	Cycles (enc)	Cycles (dec)
AES (Poettering 2013)	AVR	64	128	1570	–	2739	3579
HIGHT (Eisenbarth et al. 2012)	AVR	64	128	5672	–	2964	2964
IDEA (CEisenbarth et al. 2012)	AVR	64	80	596	–	2700	15,393
KATAN (Eisenbarth et al. 2012)	AVR	64	80	338	18	72,063	88,525
KLEIN (Eisenbarth et al. 2012)	AVR	64	80	1268	18	6095	7658
PRESENT (Eisenbarth et al. 2012)	AVR	64	128	1000	18	11,342	13,599
TEA (Eisenbarth et al. 2012)	AVR	64	128	648	24	7408	7539
PRINCE (Koo et al. 2008)	AVR	64	128	1574	24	3253	3293
SKIPJACK (Eisenbarth et al. 2007)	Power TOSSIM	64	80	5230	328	17,390	–
RC5 (Eisenbarth et al. 2007)	Power TOSSIM	64	128	3288	72	70,700	–
GRC	Zynq-7000	64	64	826	30,516	2568	2972

Cryptanalysis

In the SCESW scheme, the data is generated with the H function and the master key *key*. If the master key *key* is not disclosed to an unauthorized user, then the unauthorized user cannot get the data from the H function only through the tag information and the secret message. Access to sensitive information in the original plaintext.

Related keys

The attacker can complete the operation of attacking the user by performing decryption operations using unknown or partially previously known keys. Related key attacks mainly rely on the slow diffusion or symmetry in the key expansion block. The key expansion process of the GRC algorithm is designed to quickly and nonlinearly diffuse the difference of cipher keys to round keys.

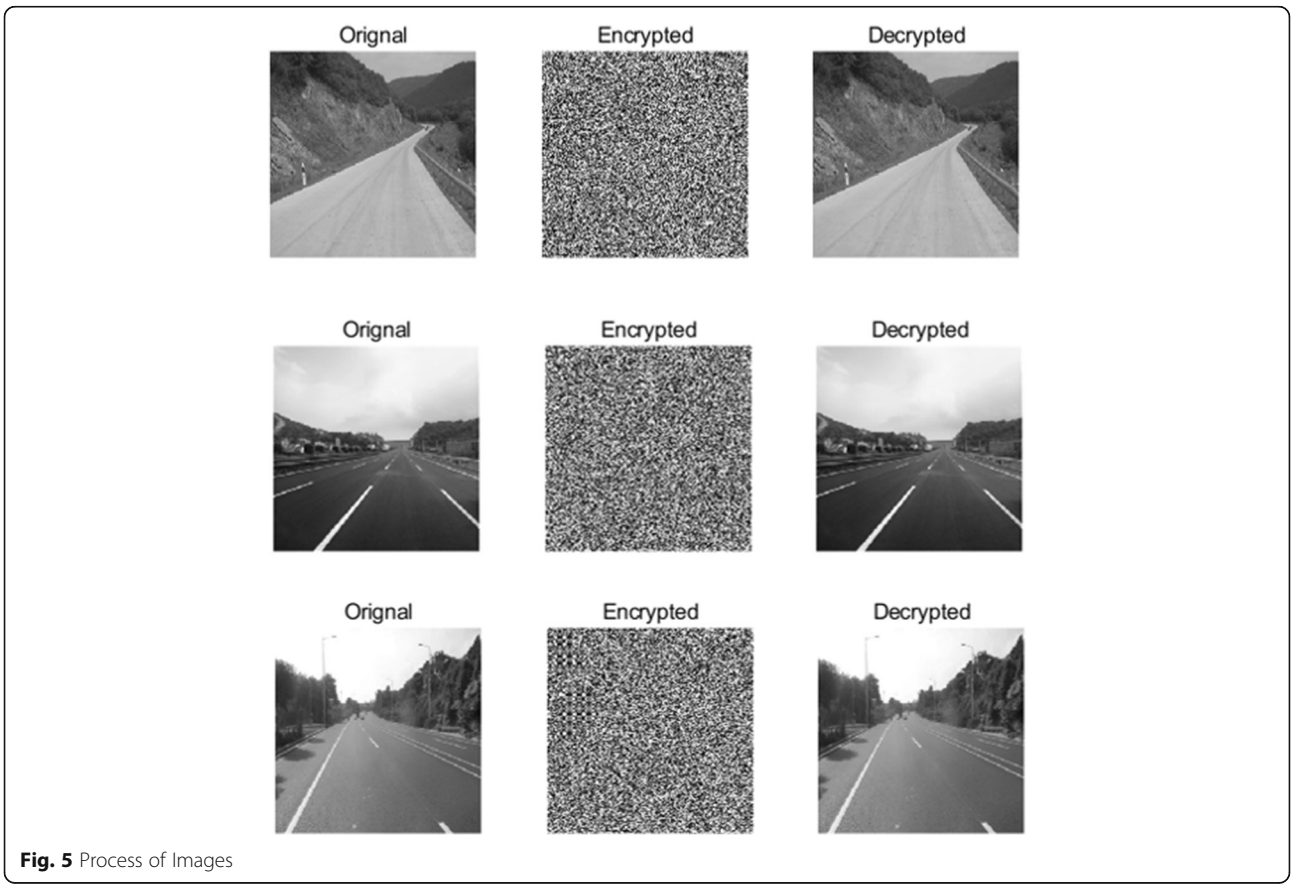

Fig. 5 Process of Images

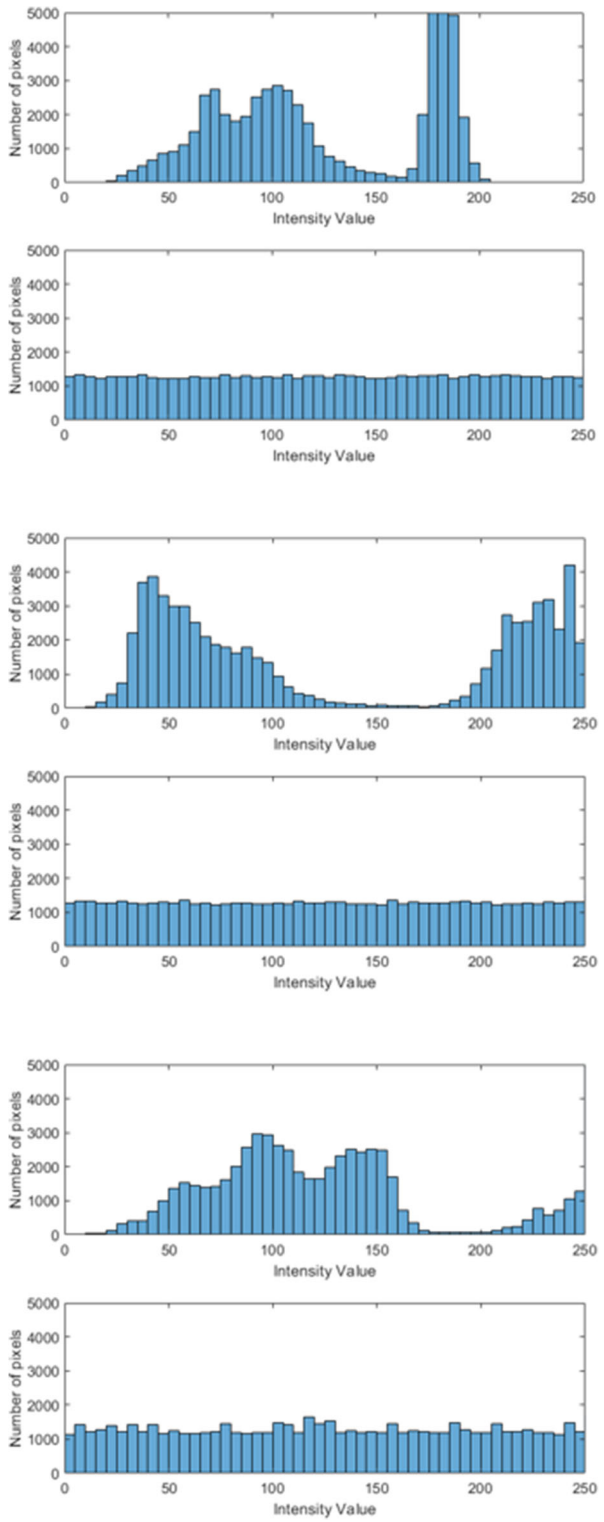

Fig. 6 Histogram Results

Performance analysis

We performed our GRC algorithm simulation using MATLAB on a computer with an Intel Core i7–7700@2.8Ghz processor. In order to make our GRC algorithm meaningful, we performed it on the Xilinx VIVADO platform on the FPGA algorithm design and simulation, and after all this was verified successfully, we ported the GRC algorithm to Xilinx. Real-world validation on the Zynq-7000 platform with no failures in operation. It is found that the GRC algorithm takes only 0.002 milliseconds to encrypt on Zynq-7000, which corresponds to the following With a decryption time of 0.003 milliseconds, the Zynq-7000 platform takes up only about 30 kilobytes memory size in the process. Finally, we compared the GRC algorithm with other algorithms deployed on the hardware platform and the results are shown in the Table 4 below, we can clearly see that the GRC algorithm we have designed greatly reduces the number of encryption and decryption rounds compared to other algorithms, while enhancing security.

The results in Fig. 5 show that accurate decryption of an image is only possible if the correct key is used to decrypt the image, otherwise the image will still not Identify. Further in the histogram results of the original and encrypted images in Fig. 6 the uniform distribution of the encrypted intensity is a representation of the desired safety. A maximum entropy of 8 bits can be achieved for the grayscale images. As can be seen from the results in Table 5, the entropy of all encrypted images is close to the maximum value, depicting a property of the algorithm. The final correlation in Fig. 7 illustrates the comparison between the raw and encrypted data. The raw data, which in our case is an image, can be seen to be highly correlated and leaves a high value of the correlation coefficient. And the encrypted images have little to no correlation.

Conclusion

In this highly prosperous era of information technology, the Internet of Things has been integrated into our daily lives. Numerous devices are constantly communicating with each other, and there are bound to be problems with the secure transmission of information during the communication process.

To this end, this paper proposes a new safety algorithm, named GRC, using automated driving as an example. We have implemented this algorithm on different hardware platforms, thus making it an optional solution for IoT applications. Next, we are interested in

Table 5 Result for Correlation and Entropy

Image	Size	Correlation		Entropy	
		Original	Encrypted	Original	Encrypted
Scene 1	512*512	0.9165	0.0016	7.0106	7.9981
Scene 2	512*512	0.8158	0.0023	7.2516	7.9985
Scene 3	512*512	0.9021	0.0018	7.4329	7.9979

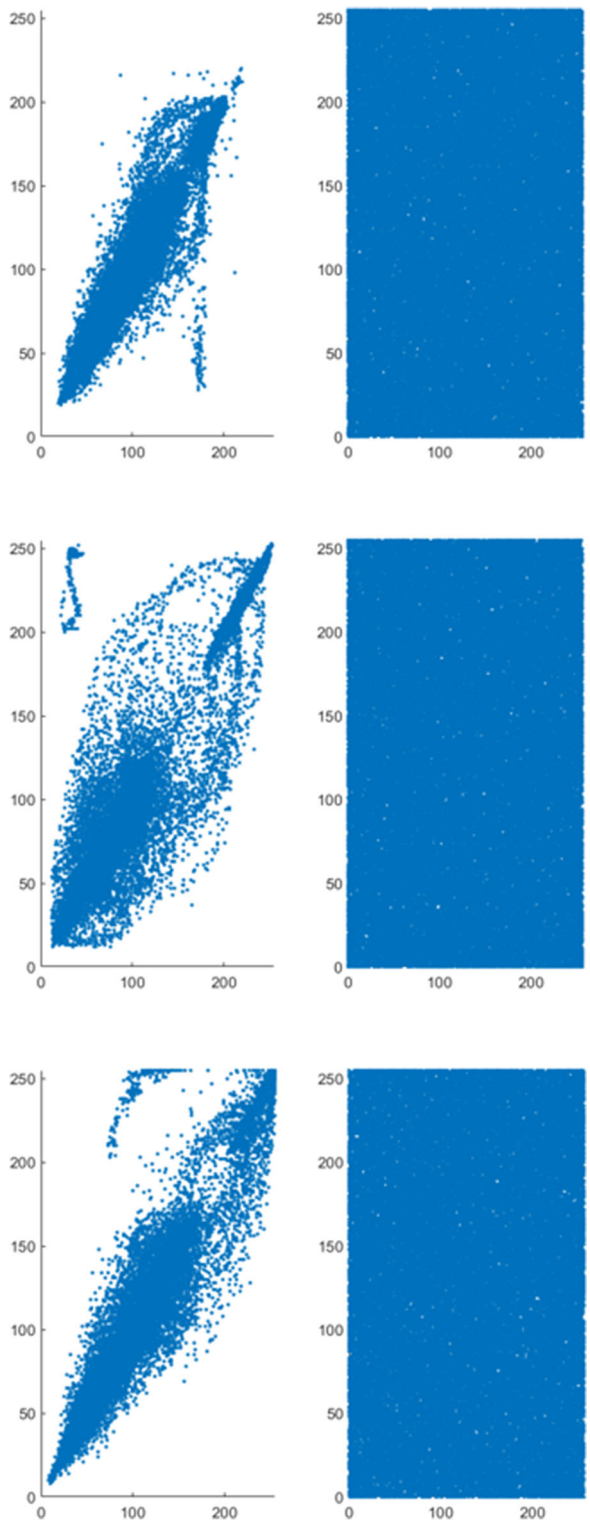

Fig. 7 Correlation Results

performing a detailed performance evaluation of this algorithm in more scenarios to address the multiple possible future attacks.

Authors' contributions
The algorithm design and writing of the manuscript was completed by Runchen Gao. The contribution of Shen Li was his reasonable analysis of algorithm and the solution of the problems during algorithm was done by Rui Guo. Yuqi Gao verified the integrity of algorithm, completed flow charts and typeset our manuscript. The author(s) read and approved the final manuscript.

Funding
This work was supported by the National Natural Science Foundation of China under Grant 61802303, 61772418 and 61602378, the Key Research and Development Program of Shaanxi under Grant 2020ZDLGY08-04 and 2019KW-053, the Innovation Capability Support Program in Shaanxi Province of China under Grant 2020KJXX-052 and 2017KJXX-47, the Natural Science Basic Research Plan in Shaanxi Province of China under Grant 2019JQ- 866, 2018JZ6001 and 2016JM6033, the Research Program of Education Bureau of Shaanxi Province under Grant 19JK0803, the New Star Team of Xi'an University of Posts and Telecommunications under Grant 2016-02.

Availability of data and materials
Not applicable.

Competing interests
Peace and Love.

Received: 21 August 2020 Accepted: 15 December 2020
Published online: 01 February 2021

References

Chandramouli R, Bapatla S, Subbalakshmi K, Uma R (2006) Battery power-aware encryption. ACM Trans Inf Syst Security (TISSEC) 9(2):162–180. https://doi.org/10.1145/1151414.1151417

Eisenbarth T. et al. (2012) Compact Implementation and Performance Evaluation of Block Ciphers in ATtiny Devices. In: Mitrokotsa A., Vaudenay S. (eds) Progress in Cryptology - AFRICACRYPT 2012. AFRICACRYPT 2012. Lecture Notes in Computer Science, vol 7374. Springer, Berlin. https://doi.org/10.1007/978-3-642-31410-0_11

Coppersmith D (1994) The data encryption standard (DES) and its strength against attacks. IBM J Res Dev 38(3):243–250. https://doi.org/10.1147/rd.383.0243

Daemen J (1995) Cipher and hash function design strategies based on linear and differential cryptanalysis. In: Ph.D. dissertation, Doctoral Dissertation, March 1995, KU Leuven

Ebrahim M, Chong CW (2013) Secure Force: A low-complexity cryptographic algorithm for Wireless Sensor Network (WSN). In: 2013 IEEE International Conference on Control System, Computing and Engineering, Mindeb, pp 557–562. https://doi.org/10.1007/978-3-642-31410-0_11

Eisenbarth T, Kumar S, Paar C, Poschmann A, Uhsadel L (2007) A survey of lightweight-cryptography implementations. IEEE Design Test Comput 24(6):522–533. https://doi.org/10.1109/MDT.2007.178

Koo WK, Lee H, Kim YH, Lee DH (2008) Implementation and Analysis of New Lightweight Cryptographic Algorithm Suitable for Wireless Sensor Networks. In: 2008 International conference on information security and assurance (isa 2008), Busan, pp 73–76. https://doi.org/10.1109/ISA.2008.53

Meng Q, Ma J, Chen K, Miao Y (2018) J Commun 39(4):167–175

Muhammad U, Irfan AM (2017) Imran. Int J Adv Computr Sci Appl 8(1):402–411

Poettering B (2013) Rijndaelfurious aes-128 implementation for avr devices (2007)

Publisher's Note