Computer Modeling in
Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2022.022462

ARTICLE

Certrust: An SDN-Based Framework for the Trust of Certificates against Crossfire Attacks in IoT Scenarios

Lei Yan[1], **Maode Ma**[2], **Dandan Li**[1], **Xiaohong Huang**[1,*], **Yan Ma**[1] **and Kun Xie**[1]

[1]School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications, Beijing, China

[2]College of Engineering, Qatar University, Doha, Qatar

*Corresponding Author: Xiaohong Huang. Email: huangxh@bupt.edu.cn

Received: 10 March 2022 Accepted: 12 May 2022

ABSTRACT

The low-intensity attack flows used by Crossfire attacks are hard to distinguish from legitimate flows. Traditional methods to identify the malicious flows in Crossfire attacks are rerouting, which is based on statistics. In these existing mechanisms, the identification of malicious flows depends on the IP address. However, the IP address is easy to be changed by attacks. Compared with the IP address, the certificate is more challenging to be tampered with or forged. Moreover, the traffic trend in the network is towards encryption. The certificates are popularly utilized by IoT devices for authentication in encryption protocols. DTLShps proposed a new way to verify certificates for resource-constrained IoT devices by using the SDN controller. Based on DTLShps, the SDN controller can collect statistics on certificates. In this paper, we propose Certrust, a framework based on the trust of certificates, to mitigate the Crossfire attack by using SDN for IoT. Our goal is threefold. First, the trust model is built based on the Bayesian trust system with the statistics on the participation of certificates in each Crossfire attack. Moreover, the forgetting curve is utilized instead of the traditional decay method in the Bayesian trust system for achieving a moderate decay rate. Second, for detecting the Crossfire attack accurately, a method based on graph connectivity is proposed. Third, several trust-based routing principles are proposed to mitigate the Crossfire attack. These principles can also encourage users to use certificates in communication. The performance evaluation shows that Certrust is more effective in mitigating the Crossfire attack than the traditional rerouting schemes. Moreover, our trust model has a more appropriate decay rate than the traditional methods.

KEYWORDS

Trust model; certificate; SDN; Crossfire attack; bayesian trust system; forgetting curve; IoT

1 Introduction

With the rapid development of the Internet of Things (IoT), the large volume, pervasiveness, and high vulnerability of IoT devices have attracted many bad actors, particularly those orchestrating distributed denial-of-service (DDoS) attacks [1]. The emergence of Software-Defined Networking (SDN) offers several new advantages to defend against conventional DDoS attacks [2]. The global network view and dynamic network policy update in SDN make the traffic characteristics analysis easy and the DDoS attack prevention timely. The improvement of the defense stimulates the evolution of the attack. The attack traffic is inclined to mimic the legal traffic behaviors. Moreover, this mimicking leads to the invalidation of the traffic characteristics analysis. The Crossfire attack is such a sophisticated and powerful link-flooding attack. This attack cuts off network connections to a target area by flooding low-intensity flows on the selected links around the target area [3]. These low-intensity flows are hard to distinguish from legitimate flows.

Before flooding low-intensity flows to the target links, the attackers need to determine the target links by traceroute. The moving target defense based on traceroute packets analysis can prevent the attackers from finding out the target links [4]. Meanwhile, the traceroute is not only used by the attackers but also utilized in network measurement. Thus, it is hard to distinguish between the legal and malicious traceroute packets. The interference of legal traceroute packets may bring trouble to the target link selection in the moving target defense. Hence, we do not choose the traceroute-based methods.

Another kind of defensive approach is to identify and block the low-intensity flows flooded by the attackers. Statistics-based methods are effective in identifying the sophisticated malicious flows of the Crossfire attack. Traffic rerouting is a classical measure to conduct the repeated statistical experiments [5–7]. After several traffic rerouting rounds, the malicious flows can be identified as the flows with their source IP address always appearing in each attack round. Besides rerouting, the correlation analysis based on IP addresses also can identify the malicious flows in the Crossfire attack [8].

The above identification mechanisms utilize the IP address to identify the malicious flows. Thus, by merely modifying the IP address, an adversary can avoid the identifying mechanism. Even the IP address is required to be changed in many normal scenarios, such as using IPv6 temporary address [9], IP address mutation communications [10] and Mobile IP communications [11,12]. Therefore, the IP-based identification mechanisms may no longer work well in these cases.

A certificate signed and issued by a certificate authority (CA) is much harder to be forged or tampered with than the IP address. Moreover, the certificates are popularly utilized in IoT. On the one hand, in the device-to-device communication scenario, most connected devices do not previously know each other [13]. Thus, the IoT devices in such a situation usually depend on asymmetric encryption techniques and mainly utilize certificates for identifying and authenticating entities [13]. On the other hand, in the scenario of the IoT devices communicating with the cloud service providers, the traffic trend on the Internet is towards encryption. For example, the percentage of web pages loaded by Firefox using HTTPS has sustained growth in recent years [14]. This percentage has exceeded 80% since 2020 for global users. The above example illustrates that service providers on the Internet are inclined to provide encrypted connections. Thus, mimicking the legal traffic, the traffic of Crossfire attacks will have the same trend towards encryption. Furthermore, the certificate is also widely utilized by encryption protocols, such as TLS/DTLS and IPsec, for authenticating the IoT devices. Therefore, we believe that the certificate is an excellent candidate to identify the malicious flows for the Crossfire attack in the IoT scenario.

The certificate-based trust model depends on the statistics on the participation of the certificates in each Crossfire attack. Thus, we need to detect the Crossfire attack accurately. The existing detection methods are implemented by monitoring the congestion on pre-selected target links [6,7] or analyzing the traffic correlation on every link [8]. Without a judgment on whether a separate target area exists, the above detection measures fail to distinguish the Crossfire attack from other types of link-flooding attacks.

The heavy computational overhead of certificate verification makes it hard to implement the certificates on resource-constrained IoT devices. DTLShps [15] provides a new thought to simplify Datagram Transport Layer Security (DTLS) protocol for resource-constrained IoT devices by using SDN. All the handshake messages of DTLShps sessions are sent to the SDN controller by the SDN switch. The certificates will be verified in the controller instead of IoT devices. DTLShps can reduce about 89% energy consumption and 77% computational overhead on an IoT device compared with the traditional DTLS. For other security protocols, such as TLS and IPsec, the handshake messages of these protocols also can be sent to the controller. Then, let the controller verify the certificates contained in the handshake messages. Based on DTLShps, we can use the SDN controller to collect statistics on certificates.

In this paper, as our major contribution, a trust framework (Certrust) is designed to mitigate the link congestion during the Crossfire attack over SDN in the IoT scenario. The direct trust of certificates is calculated through the Bayesian trust system, which is based on the statistics on the historical behavior of certificates. The historical data in the Bayesian trust system is decayed by the forgetting curve instead of the traditional decay method [16–18] for achieving a more moderate decay rate. The recommendation trust of certificates is calculated depending on the reliability of the CA. For detecting the attack accurately, we proposed a method based on graph connectivity to detect the Crossfire attack from the link congestion events. For mitigating the Crossfire attack, several trust-based routing principles, which can encourage users to use certificates in communication, are introduced. Then these principles are implemented by the bandwidth configuration in the SDN switches. The performance evaluation shows that Certrust is more effective in mitigating the link congestion in the Crossfire attack than the rerouting schemes.

The remainder of this article is organized as follows. Section 2 briefly introduces the preliminary and network scenario. Then, the motivation and the framework of Certrust is illustrated in Section 3. Section 4 constructs the trust model of the certificate. The detection mechanism and the mitigation mechanism for the Crossfire attack are provided in Section 5. Section 6 discusses the parameters in the trust model and evaluates the performance of our framework before we conclude this article in Section 7.

2 System Model

2.1 Network Model

The network model is shown in Fig. 1. The normal IoT devices, bots and decoy servers are connected to the network through the SDN switch, which performs as an IoT broker. Several bots can congest the target links by sending low-intensity traffic to different decoy servers. If the target links are congested, the target server cannot be connected by any normal IoT devices. We assume that the SDN switches and the controller are trusted, as our threat model is for the Crossfire attack. All the handshake messages of communicational security protocols, such as DTLS, TLS and IPsec, are sent to the controller by the SDN switch, as shown in DTLShps [15]. The certificates will be verified in the controller instead of IoT devices. The Certrust service is deployed on the controller. The controller will forward the processed handshake packets to the SDN switch that can deliver the handshake packets to the destination directly. After the handshake phase, the encrypted traffic between the two hosts will not pass through the controller any more.

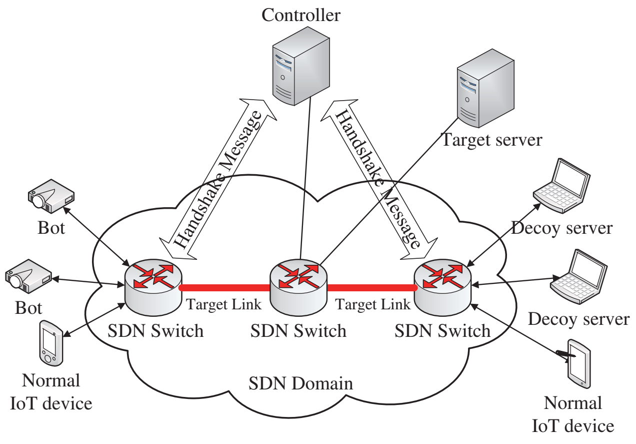


Figure 1: Network model

We study our framework in a simple SDN scenario, in which one controller exists in a single SDN domain. The simple SDN scenario will facilitate the analysis and discussion of our framework. Moreover, it is easy to extend one controller by multi-controllers for scalability in practice. The extension for multi-controllers in a single SDN domain has been illustrated in [15]. The cooperation between multiple SDN domains is shown in [19].

2.2 Threat Model

Our study focuses on the Crossfire attack, a sophisticated link flood attack. The Crossfire attack can degrade and even cut off network connections to a variety of selected server targets by flooding a few links in the network [3]. The server targets can be selected as the servers of an enterprise, a city, a state, or a small country. A small set of bots directs low-intensity flows to a large number of publicly accessible servers (decoy servers). The concentration of these flows floods a small set of carefully chosen links and effectively disconnects selected target servers from the Internet. The sources of the Crossfire attack are undetectable by any targeted servers since they no longer receive any messages. Moreover, the low-intensity, individual flows of the Crossfire attack are indistinguishable from legitimate flows by traditional traffic anomaly detection. The attack persistence can be extended virtually indefinitely by changing bots, decoy servers, and target links while maintaining the same disconnection targets.

2.3 Bayesian Trust System

The Bayesian trust system, which is also called the Beta reputation system, works based on the beta probability density function (PDF) [20,21]. This PDF can represent probability distributions of binary (i.e., positive or negative) events. The beta PDF is indexed by two parameters α and β. The beta PDF denoted by $f(x|\alpha,\beta)$ can be expressed using the gamma function Γ as:

$$f(x|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}, \tag{1}$$

where $0 \leqslant x \leqslant 1$, $\alpha > 0$, $\beta > 0$, with the restriction that the probability variable $x \neq 0$ if $\alpha < 1$, and $x \neq 1$ if $\beta < 1$. The probability expectation value of the beta distribution is given by:

$$E(x) = \frac{\alpha}{\alpha + \beta}. \tag{2}$$

Moreover, trust is a subjective expectation a subject has about another's future behavior based on the history of their encounters [22]. Therefore, the most natural is to define the trust value, which is denoted by Tv, as $E(x)$.

After observing r positive and s negative outcomes of the binary events, the *a posteriori* distribution is the beta PDF with $\alpha = r + 1$ and $\beta = s + 1$, where $r, s > 0$. Thus,

$$Tv = E(x) = \frac{r+1}{(r+1)+(s+1)} = \frac{r+1}{r+s+2}. \tag{3}$$

Let $(\hat{T}v - \varepsilon, \hat{T}v + \varepsilon)$ be the confidence interval with the confidence degree γ of Tv, ε is the error level. γ can be expressed as:

$$\gamma = P\left(\hat{T}v - \varepsilon < Tv < \hat{T}v + \varepsilon\right) = \frac{\int_{\hat{T}v-\varepsilon}^{\hat{T}v+\varepsilon} x^{r-1}(1-x)^{s-1}dx}{\int_0^1 x^{r-1}(1-x)^{s-1}dx} = \frac{\Gamma(r)\Gamma(s)}{\Gamma(r+s)} \int_{\hat{T}v-\varepsilon}^{\hat{T}v+\varepsilon} x^{r-1}(1-x)^{s-1}dx. \tag{4}$$

The accuracy of Tv can be improved by increasing the number of samples [23]. Let γ_0 be the threshold of confidence level. When the accuracy is at an acceptable level (i.e., $\gamma \geqslant \gamma_0$), the Tv can be evaluated with these samples at this time. The relationship between the number of samples n_0, ε and γ_0 can be modeled as follows:

$$n_0 \geqslant -\frac{1}{2\varepsilon^2} \ln\left(\frac{1-\gamma_0}{2}\right). \tag{5}$$

3 Design of the Certrust Framework

3.1 Motivation

Traditional measures to identify malicious flows against Crossfire attacks is to use the IP address [5–8]. The IP address can be modified easily by an attacker. Thus, the detection mechanism of malicious flows will fail. Moreover, in many scenarios, the IP address even is required to be changed frequently.

On the contrary, the certificate is much harder to be tampered compared with the IP address. The certificate is based on asymmetric cryptography and signed by the secret key of a CA. A tampered certificate can be detected by the verification with the public key of the CA. Furthermore, the traffic in the network has a trend towards encryption and certificates are used widely by IoT devices. Therefore, we propose a framework to utilize the certificate to identify the malicious flows effectively.

3.2 Framework of Certrust

The framework of Certrust is shown in Fig. 2. The modules of the framework are deployed in the controller and divided into two types: packets handling and trust handling. There are two packets handling modules: *certificate verification* and *trust-based routing policy generation*. The packets handling modules are responsible for dealing with the packet-in messages sent from the SDN switches. When a certificate is sent to the controller by an SDN switch, the certificate will be verified by the *certificate verification* module firstly. After passing the verification, the *trust-based routing policy generation* module will generate a routing policy depending on the trust level of the certificate. Then, the routing policy will be sent to the relevant SDN switches.

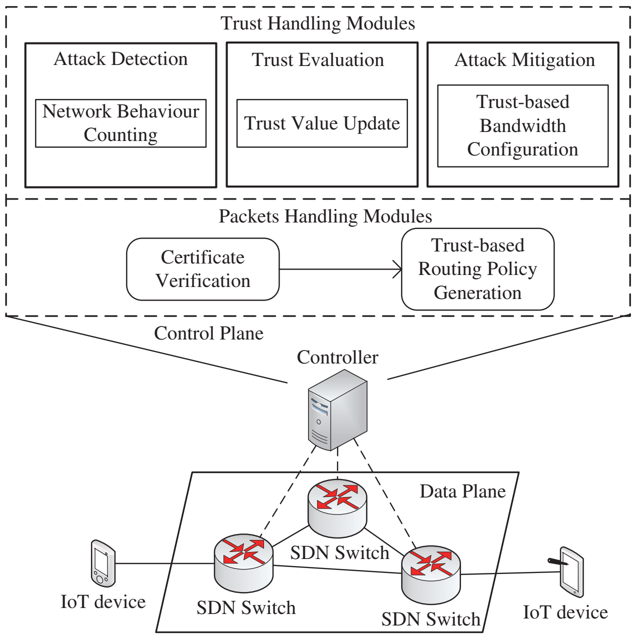


Figure 2: Framework of certrust

Moreover, three trust handling modules provide the trust-related data required by the packets handling modules: *network behavior counting*, *trust value update*, and *trust-based bandwidth configuration*. The *network behavior counting* module is designed to detect the Crossfire attacks and record the network behaviors of the certificates during the Crossfire attacks. The *trust value update* module maintains the trust model based on the statistics on the network behaviors of the certificates. The *trust-based bandwidth configuration* module is responsible for grading the certificates according to their trust value and configuring the bandwidths of each trust level on the SDN switches for mitigating the Crossfire attack.

3.2.1 Certificate Verification Module

This module is implemented by the certificate verification mechanism proposed in [15]. It is easy to extend this progress to other security protocols. The handshake process in [15] is shown in Fig. 3. The IoT device that initiates the session performs as a client, and the IoT device that responds to the session performs as a server. The client and server send their certificate chain to the controller by the *Certificate* message. After the controller has completed the certificate verification, the controller sends the identity information in the certificate to the corresponding end by the *Identity* message.

To deploy certificate-based authentication on resource-constrained IoT devices, it is common to shift the certificate verification to a trusted third party, such as a delegation server [24–26]. The controller performs as the trusted third party in our framework. Moreover, the controller needs to encourage the client to use the certificate, as our framework is based on the trust model of certificates. Thus, we make some modifications to the above handshake process. The controller will always send the *CertificateRequest* message to request the client's certificate, regardless of whether the server has sent the *CertificateRequest* message to the controller. If the server has not sent the *CertificateRequest* message to the controller, the controller will not send the *Identity* message to the server to inform the client's identity. Otherwise, the controller will send. If the client refuses to send its certificate, the client will send a *Certificate* message with no certificate to the controller. Then the controller will send an *Identity* message with no information to the server if the server has sent the *CertificateRequest* message to the controller.

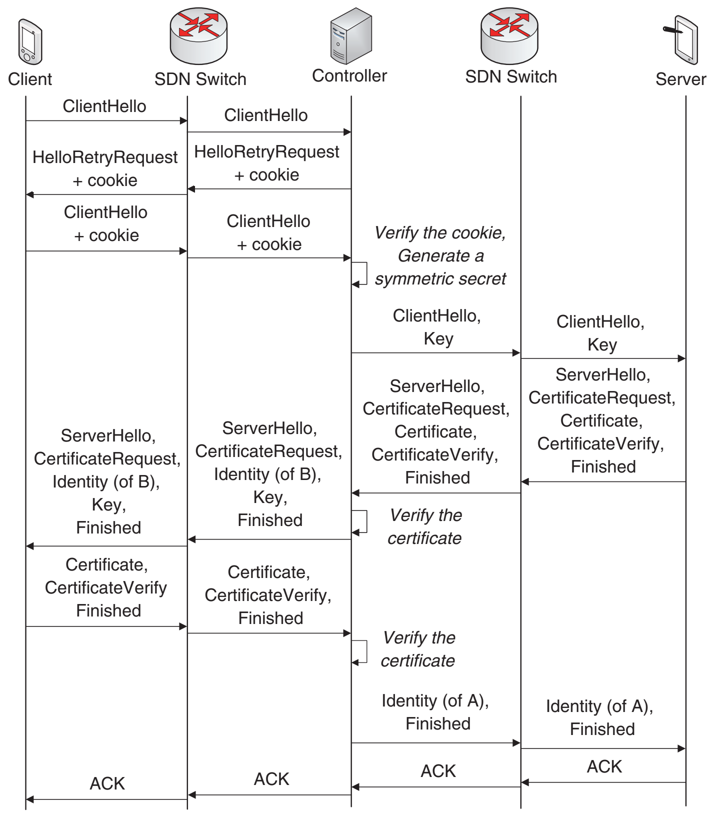

Client
SDN Switch
Controller
SDN Switch
Server
ClientHello
ClientHello
HelloRetryRequest + cookie
HelloRetryRequest + cookie
ClientHello + cookie
ClientHello + cookie
Verify the cookie, Generate a symmetric secret
ClientHello, Key
ClientHello, Key
ServerHello, CertificateRequest, Certificate, CertificateVerify, Finished
ServerHello, CertificateRequest, Certificate, CertificateVerify, Finished
ServerHello, CertificateRequest, Identity (of B), Key, Finished
ServerHello, CertificateRequest, Identity (of B), Key, Finished
Verify the certificate
Certificate, CertificateVerify Finished
Certificate, CertificateVerify, Finished
Verify the certificate
Identity (of A), Finished
Identity (of A), Finished
ACK
ACK
ACK
ACK

Figure 3: Flow diagram of handshake in DTLShps

The user, who wants to be anonymous, can use the anonymous certificate, as presented in [27]. The anonymous certificate is a syntactically valid X.509 certificate, in which the *Subject* field contains a pseudonym. The anonymous certificate also can accumulate trust in our framework.

3.2.2 Trust-Based Routing Policy Generation Module

After verifying a certificate, the controller will inquire about the trust level of the certificate. The trust level of certificates is recorded in the controller by the *trust-based bandwidth configuration* module and obtained by searching the records. The controller will generate and issue the routing policy to the relevant SDN switches depending on the trust level of the certificate. Several queues have been created on each egress port of the SDN switches. Each queue corresponds to a trust level. Different queues have been configured with different bandwidths by the *trust-based bandwidth configuration* module. The routing policy issued from the controller will make the traffic from different trust levels handled by the corresponding queue on the SDN switches. This module is also responsible for recording the mapping between the certificate and the flow in the log file.

3.2.3 Network Behavior Counting Module

This module firstly uses the detection mechanism proposed in [6] to detect link congestion. The available bandwidth of the links over the whole network is monitored by the SDN controller. If the available bandwidth of a link is exhausted, link congestion occurs. The controller will record the period when the congestion lasted and the locations (SDN switches) where the congestion happened. Secondly, this module collects the information of the flows and certificates participating in the link congestion from the log files recorded by the *trust-based routing policy generation* module. Thirdly, the Crossfire attack is detected from the link congestion events by a method based on the graph connectivity. Finally, if a Crossfire attack is detected, the count of the network behaviors of relevant certificates is updated.

3.2.4 Trust Value Update Module

This module updates the trust value of the certificate based on the network behavior and activity of the certificate. The statistics for network behavior is made by the *network behavior counting* module. Moreover, the activity can be calculated through the log files recorded by the *trust-based routing policy generation* module.

3.2.5 Trust-Based Bandwidth Configuration Module

The trust level is graded depending on the trust values maintained by the *trust value update* module. Different maximum and minimum available bandwidths of the queues on the SDN switches are set according to different trust levels for providing differentiated services.

4 Trust Model of Certificates

The trust of a user's certificate is an expectation about the certificate owner will act as a benign user. The user's certificate trust T_C is comprised of the direct trust T_{C-d} and the recommendation trust T_{C-r}. T_{C-d} depends on the network behavior of the certificate's owner. Considering that the authenticity of a certificate is guaranteed by the signature of its CA, T_{C-r} is based on the trust of the CA issued that certificate. Thus, T_C is modeled as follows:

$$T_C = \omega_1 \times T_{C-d} + (1 - \omega_1) \times T_{C-r}, \tag{6}$$

where ω_1 is the weight of the direct trust of the user's certificate, $\omega_1 \in [0, 1]$.

4.1 The Direct Trust of a User's Certificate

The calculation of T_{C-d} is based on the Bayesian trust system. The historical data on network behavior of all active users will be counted whenever a Crossfire attack is reported. The active user is defined as the user who transmitted data through the network in several minutes or hours around the point that the Crossfire attack occurred. The granularity of time can be set according to the different security levels. If the traffic from one user's certificate participated in the link congestion of a Crossfire attack, the count of suspicious actions of that certificate v is added by one. If the traffic from one user's certificate did not participate in the link congestion of a Crossfire attack, the count of legal actions of that certificate u is added by one. Hence, after the Crossfire attack has happened $u + v$ times, according to (3), T_{C-d} can be expressed as

$$T_{C-d} = \frac{u + 1}{u + v + 1}. \tag{7}$$

It is reasonable to suspect the owner of a certificate, whose traffic often attended the link congestion caused by the Crossfire attacks, is an attacker.

We further consider the factor of time in our model. The more recent the historical data on network behavior is, the more valuable it is to evaluate the trust level of certificates. Time is divided into time slots according to a fixed period T. T can be several hours or days. A weight η is utilized to decay the historical data in each time slot. Thus, after the nth time slot, the total count of legal actions $u(n)$ and suspicious actions $v(n)$ for a certificate can be expressed as follows:

$$u(n) = \sum_{i=1}^{n} u_i \times \eta_i, \tag{8}$$

$$v(n) = \sum_{i=1}^{n} v_i \times \eta_i, \tag{9}$$

where

u_i is the count of legal actions in the ith time slot,

v_i is the count of suspicious actions in the ith time slot,

η_i is the decay weight of the ith time slot,

n is the sequence number of the current time slot,

"1" is the sequence number of the initial time slot.

Moreover, a forgetting curve shows the nonlinear decreasing law of human memory retention [28,29]. We use human memory to simulate the historical data. Thus, the forgetting curve can be applied to obtain the decay weight η. An exponential function with base e is chosen to model the forgetting curve. Therefore, the weight η_i is expressed as follows:

$$\eta_i = e^{-\frac{\Delta t}{\lambda_1}}, \tag{10}$$

where Δt is the period from the ith time slot to the current time slot, λ_1 is the decay rate parameter. Thus, (8) and (9) can be updated as

$$u(n) = \sum_{i=1}^{n} u_i \times e^{-\frac{\Delta t}{\lambda_1}} = \sum_{i=1}^{n} u_i \times e^{-\frac{n-i}{\lambda_1}}, \tag{11}$$

$$v(n) = \sum_{i=1}^{n} v_i \times e^{-\frac{\Delta t}{\lambda_1}} = \sum_{i=1}^{n} v_i \times e^{-\frac{n-i}{\lambda_1}}, \tag{12}$$

Consequently, (7) can be updated as follows:

$$T_{C-d} = \frac{u(n)+1}{u(n)+v(n)+2} = \frac{\sum_{i=1}^{n} u_i \times e^{-\frac{n-i}{\lambda_1}} + 1}{\sum_{i=1}^{n} u_i \times e^{-\frac{n-i}{\lambda_1}} + \sum_{i=1}^{n} v_i \times e^{-\frac{n-i}{\lambda_1}} + 2}. \tag{13}$$

Only if the number of samples is equal to or greater than the threshold in (5), the direct trust of the user's certificate can be calculated by (13). If the number of samples is less than the threshold, T_C is directly equal to T_{C-r}. This measure can encourage users to choose a CA with a good reputation, as T_{C-r} is based on the trust of the CA's certificate.

4.2 The Recommendation Trust of a User's Certificate

When a CA signs a certificate with its private key, it means that the CA uses its reputation to guarantee that the public key in the certificate belongs to the certificate owner. Thus, the recommendation trust of a certificate is related to its CA's reputation. A user's certificate is generally contained in a certificate chain, as shown in Fig. 4. The owner's identity and public key in a user's certificate are signed by the private key of its issuer (CA_1). The public key of CA_1, which is contained in CA_1's certificate, can be used to verify that the certificate issued by CA_1 is trustworthy. Furthermore, CA_1's certificate is also needed to be verified by the public key of its issuer (CA_2), which is contained in CA_2's certificate. This verification cycle continues until the *root CA*, which self-signs its public key with its private key. These certificates from the user's certificate to the root certificate constitute a certificate chain.

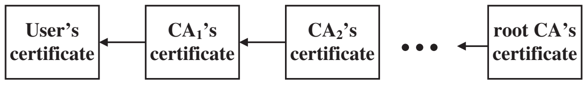


Figure 4: The schematic diagram of a certificate chain

The closer a CA's certificate is to the user's certificate in a certificate chain, the tighter the correlation between them is. Therefore, T_{C-r} is calculated based on the trust of all the CA's certificates in the certificate chain. We assume that there are m CA's certificates in a certificate chain, and the sequence number of the root CA's certificate is m. Thus, T_{C-r} is modeled as

$$T_{C-r} = \frac{\sum_{i=1}^{m} \xi_i \times T_{CAi}}{\sum_{i=1}^{m} \xi_i}, \tag{14}$$

where T_{CAi} is the trust of the ith CA's certificate in the certificate chain, $T_{CAi} \in [0, 1]$; ξ_i is the weight of the trust of the ith CA's certificate in the certificate chain, $\xi_i \in [0, 1]$. Moreover, ξ_i can be calculated as

$$\begin{cases} \xi_1 & = & 1, & \\ \xi_{i+1} & = & \xi_i - \Delta, & if\ \xi_i - \Delta \geqslant 0 \\ \xi_{i+1} & = & 0, & if\ \xi_i - \Delta < 0 \end{cases} \tag{15}$$

where Δ is the decay factor for attenuating the ξ_i, $\Delta \in (0, 1)$. If there are too many CA's certificates in a certificate chain, the trust of those CA's certificates, which are too far away from the user's certificate, can be ignored. For example, when $\Delta = 0.1$, $\xi_{11} = \xi_{10} - 0.1 = 0$. Thus, the trust of all the CA's certificates whose sequence number greater than ten will be ignored.

4.3 The Trust of a CA's Certificate

The trust of a CA's certificate T_{CA} is also comprised of the direct trust T_{CA-d} and the recommendation trust T_{CA-r}, like the trust of a user's certificate. Thus, T_{CA} is modeled as

$$T_{CA} = \omega_2 \times T_{CA-d} + (1 - \omega_2) \times T_{CA-r}, \tag{16}$$

where ω_2 is the weight of the direct trust of the CA's certificate, $\omega_2 \in [0, 1]$.

T_{CA-d} hinges on the average trust value of all the certificates issued by this CA. Assuming one CA has issued k certificates, T_{CA-d} is modeled as follows:

$$T_{CA-d} = \frac{\sum_{i=1}^{k} T_{C-di}}{k}. \tag{17}$$

where T_{C-di} is the direct trust value of the ith certificate issued by the CA.

T_{CA-r} rests on the evaluation of the CA coming from some credible websites, for example, the PageRank value PR of a CA's website from Google. The PageRank value is between 0 and 10. Thus, T_{CA-r} can be expressed as

$$T_{CA-r} = \frac{PR}{10}. \tag{18}$$

Before the first certificate issued by a CA appears in the network, T_{CA} is directly equal to T_{CA-r}. If a CA is brand new so that there is on recommendation trust of the CA's certificate, the trust value of this CA's certificate is set to a medium value of 0.5.

4.4 Parameter Analysis

4.4.1 ω_1

ω_1, which is the weight of T_{C-d}, is calculated based on the degree of the user activity, for T_{C-d} depends on the user's behavior. The higher the degree of the user activity is, the more proportion T_{C-d} will take in the calculation of T_C. The user activity is defined as the sum of a certificate's active counts in the last n' time slots. The period of one time slots is set as a fixed value T′. For example, the active count in every time slot can be the number of active days in every month or the number of active hours in every day.

The time factor is also considered in the model of ω_1. The more recent the data of the active count is, the more valuable it is to calculate the degree of the user activity. Similar to T_{C-d}, a forgetting curve, which is also modeled by an exponential function with base e, is utilized to decay the active count in different time slots. The decay weight φ_i of the ith time slot can be expressed as

$$\varphi_i = e^{-\frac{\Delta t}{\lambda_2}}, \tag{19}$$

where Δt is the period from the ith time slot to the current time slot, λ_2 is the decay rate parameter. Therefore, ω_1 is modeled as follows:

$$\omega_1 = \frac{\sum_{i=1}^{n'} A_i \times \varphi_i}{n' \times \max(A_i)} = \frac{\sum_{i=1}^{n'} A_i \times e^{-\frac{\Delta t}{\lambda_2}}}{n' \times \max(A_i)} = \frac{\sum_{i=1}^{n'} A_i \times e^{-\frac{n'-i}{\lambda_2}}}{n' \times \max(A_i)}, \tag{20}$$

where

A_i is the active count of the user's certificate in the ith time slot,

n' is the sequence number of the current time slot,

"1" is the sequence number of the initial time slot.

Moreover, the parameter λ_2 can be defined as

$$\lambda_2 = \frac{t_0}{\ln 2}, \tag{21}$$

where t_0 is the half-life of the active count, after which the active count will be attenuated by half [30], $t_0 > 0$.

4.4.2 ω_2

The calculation of ω_2, which is the weight of T_{CA-d}, depends on the activity of the CA. The activity of a CA is defined as the average user activity of all the certificates issued by the CA. Assuming a CA has issued k certificates, the active count of the CA's certificate in the ith time slot A_{CAi} is modeled as follows:

$$A_{CAi} = \frac{\sum_{j=1}^{k} A_{ij}}{k}, \tag{22}$$

where A_{ij} is the active count of the jth certificate issued by the CA in the ith time slot. Thus, the activity of the CA can be calculated as the sum of active counts of the CA's certificate in the last n' time slots.

The model of ω_2 is similar to that of ω_1. The forgetting curve, which is modeled by the same exponential function as in (19), is also utilized to decay the active count of the CA's certificate in each time slot. Therefore, ω_2 is modeled as follows:

$$\omega_2 = \frac{\sum_{i=1}^{n'} A_{CAi} \times \varphi_i}{n' \times \max\left(A_{CAi}\right)} = \frac{\sum_{i=1}^{n'} \frac{\sum_{j=1}^{k} A_{ij}}{k} \times e^{-\frac{\Delta t}{\lambda_2}}}{n' \times \max\left(A_{CAi}\right)} = \frac{\sum_{i=1}^{n'} \sum_{j=1}^{k} A_{ij} \times e^{-\frac{n'-i}{\lambda_2}}}{n' \times k \times \max\left(A_{CAi}\right)}, \tag{23}$$

where the definitions of n' and "1" are the same as those in (20).

4.5 Trust Value Update

The trust value in the trust model is updated at the end of each time slot. Meanwhile, the period of a single time slot can be different for counting the network behavior and user activity. Thus, T_{C-d} is updated at a frequency of $1/T$, while ω_1 and ω_2 are updated at a frequency of $1/T'$. The update of any of T_{C-d}, ω_1 or ω_2 will result in the update of T_{C-r}, T_{CA} and T_C. The updated trust value will be recorded in the controller.

The count of network behaviors is updated by the *network behavior counting* module, as shown in Section 5.2. The user activity can be calculated through the log files recorded by the *trust-based routing policy generation* module, as illustrated in Section 3.2.2. For example, if a certificate occurred in the record of a specific day, the certificate can be counted as active on this day. Alternatively, if the number of occurrences of a certificate exceeded a certain threshold in one day, the certificate also can be counted as active on this day.

4.6 Time Complexity Analysis

The number of cycles of calculating ω_2 is $n' \times k$, T_{CA-d} is k, and T_{CA-r} is 1. Thus, the maximum number of cycles of calculating T_{CA} is $n' \times k$ according to (16), and T_{C-r} is $m \times n' \times k$ derived from (14). Moreover, the maximum number of cycles of calculating T_{C-d} is n, and ω_1 is n'. Therefore, the time complexity of the calculation of T_C is $O(m \times n' \times k + n)$ according to (6).

5 Attack Detection and Mitigation

5.1 The Crossfire Attack Detection

The Crossfire attack works through the link congestion. Thus, firstly it is required to record the link congestion events that occurred in the network. Secondly, the Crossfire attack is required to be detected from the link congestion events.

The link congestion in SDN can be detected by exploiting the Floodlight controller's statistics measurement APIs [6]. The link congestion can be identified by a decrease in the available bandwidth. The threshold for the available bandwidth is 10% of the total bandwidth, as set in [6]. After congestion is detected, the controller will record the period when the congestion lasted and the location (SDN switches) where the congestion happened.

We propose a method based on graph connectivity to detect the Crossfire attack from the link congestion events. A striking feature of the Crossfire attack is that all the network connections to a target area are cut off during the attack [3]. As shown in Fig. 5, after all the hollow links are congested, the network inside the broken circle loses the connection from outside during a Crossfire attack. If no area has been disconnected, the attack should be counted as a normal link-flooding attack. Otherwise, if just one link was congested, the attack should be counted as a Coremelt attack [31].

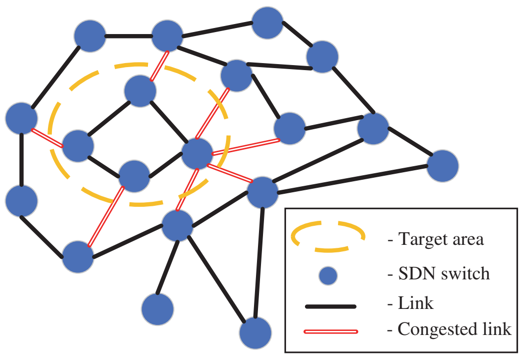


Figure 5: A demonstration of the link congestion constructing a crossfire attack

The routing topology as an undirected graph must be connected, because the unreachable routers will not present on the routing topology. The routing topology is stored in the controller. Then, the congested links in the same period are removed from the routing topology. If the rest of the routing topology is not a connected graph, there must be a certain area disconnected. Otherwise, there is no area disconnected.

The steps for detecting the Crossfire attack from the link congestion events are illustrated in Algorithm 1. The graph of routing topology $G = (S, L)$, which constructed by the switches set S and links set L, and the set of congested links in the same period Lc are taken as the input of Algorithm 1. Initially, the set Lc are removed from G. Secondly, the graph traversal algorithm breadth-first search (BFS) [32] or depth-first search (DFS) [32] is run on G and started with an arbitrary s in the set S. If every s in the set S has been explored by the graph traversal algorithm, G is connected. Thus, Algorithm 1 terminates returning the link-flooding attack type "*Normal*". Otherwise, if $|Lc|$, the cardinality of the set Lc, equals one, there is only one link congested. Therefore, Algorithm 1 ends with reporting the link-flooding attack type "*Coremelt*". In addition to the above two cases, the link-flooding attack type "*Crossfire*" is returned.

The time complexity of Algorithm 1 is the same as that of the BFS or DFS algorithm. The time complexity of the BFS and DFS algorithm both are $O(|S| + |L - Lc|)$, where

$|S|$ is the number of switches in the topological graph,

$|L - Lc|$ is the number of uncongested links in the topological graph.

Algorithm 1: Detecting the Crossfire attack from the congestion events

```
Input: Switches set $S$, links set $L$, graph $G = (S,L)$, congested links set $Lc \subseteq L$
Ouput: The attack type "Crossfire", "Coremelt" or "Normal"
1: Let $L \leftarrow L - Lc$
2: Run BFS or DFS algorithm on G, start from $\forall s \in S$
3: if every $s \in S$ has been marked as "Explored" then
4: return "Normal"
5: else if $|Lc| = 1$ then
6: return "Coremelt"
7: endif
8: return "Crossfire"
```

5.2 *Network Behavior Counting*

The network behavior counting of the certificates is based on the detection of the Crossfire attack. Whenever link congestion is detected, the controller will record the period and location of the link congestion, as shown in Section 5.1. The flows participating in the congestion can be obtained by searching the log files of the SDN switches with the period and location recorded previously. The log files can be collected by the controller from the SDN switches regularly. As controlling all the SDN switches in the network, the controller can obtain the link congestion records of the whole network. The certificates associated with the flows participating in the link congestion and active certificates in the same period can be found in the log files recorded by the *trust-based routing policy generation* module. Thus, if a Crossfire attack is detected from the link congestion events through Algorithm 1, the network behavior count of the certificates can be updated.

5.3 *The Crossfire Attack Mitigation*

The network provides differentiated services of routing according to different trust levels of certificates to mitigate the link congestion caused by the Crossfire attack. According to the trust value of the certificates, the certificates can be divided into several *high* levels, several *low* levels and one *medium* level. The traffic from the certificates in the *high* trust levels can be regarded as benign traffic. The traffic from the certificates in the *low* trust levels can be considered to be malicious traffic. The traffic from certificates in the *medium* trust level can be treated as unidentified traffic, as the application of the trust model requires accumulating a certain number of samples.

When link congestion occurs, the principles for differentiated routing services are as follows:

1. The traffic from certificates in the *high* trust levels is guaranteed to receive preferential routing.
2. The traffic from certificates in the *low* trust levels is restrained.
3. The traffic from certificates in the *medium* trust level or without a certificate is provided with a relatively fair routing service.
4. The higher trust level a certificate is in, the higher quality of network service the traffic from this certificate will enjoy in the link congestion.

These principles can encourage users to use certificates in communication. It also can encourage users to keep their certificate in the *high* trust levels and stimulate users to upgrade the trust level of their certificate. The user with a higher trust level certificate will enjoy a higher-quality network service in the link congestion. The finer the granularity of the trust level division is, the more influential the stimulation for users will be; meanwhile, the more complex the implementation will be. Thus, it needs balance. For the coarsest granularity, there should be one *high* trust level and one *low* trust level.

If the attacker does not use certificates to launch a Crossfire attack, the attack traffic will be treated as in the *medium* trust level. The attack traffic is restrained compared with the traffic from the certificate in the *high* trust levels. Moreover, the service quality of the traffic from certificates in the *high* trust levels still will be guaranteed under the attack. If the attacker uses the certificate to launch a Crossfire attack, the attack traffic will be distinguished by our trust model.

There are many other kinds of network attacks also conducted by massive traffic, such as Coremelt attacks, scanning attacks, the traditional DDoS attacks targeting a specified server and any other kinds of DDoS attacks. Most of these attacks do not use certificates. Thus, the traffic from these attacks will also be restrained as the traffic from certificates in the *medium* trust level. The traffic from certificates in the *high* trust levels can still receive a guaranteed routing service under these kinds of attacks.

We give an implementation of the trust-based routing principles based on bandwidth configuration. We take the coarsest granularity of the trust levels (just three trust levels: *high*, *medium* and *low*) as an example for a concise and explicit illustration. Three queues are created on each egress port of the SDN switches corresponding to the three trust levels. The queue's maximum and minimum available bandwidth are set according to different trust levels for providing differentiated services. For fairness, the bandwidth of each trust level is configured based on the number of certificates contained in the corresponding trust level. The routing policy issued by the *routing policy generation module* will make the traffic from different trust levels handled by the corresponding queue on the egress port of the SDN switches.

When link congestion occurs, the traffic is transmitted by the minimum available bandwidth. When the link is idle, the traffic could reach the maximum available bandwidth. The setting of the maximum available bandwidth is to encourage the users to upgrade the trust level of their certificate.

The maximum and minimum available bandwidth of each queue can be configured as the *max-rate* and *min-rate* in the SDN switch, respectively. The *max-rate* and *min-rate* for each trust level can be set according to the number of certificates in each trust level. The calculating process of the *max-rate* and *min-rate* of each trust level is shown in Algorithm 2. The inputs of Algorithm 2 are the bandwidth of an egress port of an SDN switch BW and the number of certificates in the *high*, *medium* and *low* trust level: NUM_h, NUM_m and NUM_l. The goal of Algorithm 2 is to figure out the *max-rate* and *min-rate* of the *high*, *medium* and *low* trust levels: $MaxRate_h$, $MinRate_h$, $MaxRate_m$, $MinRate_m$, $MaxRate_l$ and $MinRate_l$. Firstly, the number of certificates in the three trust levels are added up as SUM. Then, the $MaxRate_h$ is equal to BW, and $MinRate_l$ is equal to 0. Afterwards, the $MaxRate_l$ depends on the ratio of NUM_l to SUM. As the $MaxRate_m$ is required to greater than $MaxRate_l$, the $MaxRate_m$ depends on the ratio of $(NUM_l + NUM_m)$ to SUM. For the same reason, $MinRate_m$ depends on the ratio of NUM_m to $(NUM_h + 2 \times NUM_m)$, and $MinRate_h$ depends on the ratio of $(NUM_m + NUM_h)$ to $(NUM_h + 2 \times NUM_m)$. The time complexity of Algorithm 2 is $O(1)$.

Algorithm 2: Bandwidth configuration for the Crossfire attack mitigation

```
Input: The bandwidth of an egress port of an SDN switch $BW$, the number of certificates in the high, medium and low trust level $NUM_h$, $NUM_m$ and $NUM_l$
Output: The maximum and minimum bandwidth thresholds of the high, medium and low trust levels $MaxRate_h$, $MinRate_h$, $MaxRate_m$, $MinRate_m$, $MaxRate_l$ and $MinRate_l$
1: Let $SUM \leftarrow NUM_h + NUM_m + NUM_l$
2: Let $MaxRate_h \leftarrow BW$, $MinRate_l \leftarrow 0$
3: Let $MaxRate_l \leftarrow \frac{NUM_l}{SUM} \times BW$
4: Let $MaxRate_m \leftarrow \frac{NUM_m + NUM_l}{SUM} \times BW$
5: Let $MinRate_m \leftarrow \frac{NUM_m}{NUM_h + 2 \times NUM_m} \times BW$
6: Let $MinRate_h \leftarrow \frac{NUM_h + NUM_m}{NUM_h + 2 \times NUM_m} \times BW$
7: return $MaxRate_h$, $MaxRate_m$, $MaxRate_l$, $MinRate_h$, $MinRate_m$ and $MinRate_l$
```

6 Performance Evaluation

6.1 Discussion of the Parameters Selection in the Trust Model

6.1.1 λ_1

λ_1 is the decay rate factor of the certificate's action count for calculating T_{C-d}. The number of samples n_0 is required to no less than a threshold, which is shown in (5). Applying $\gamma_0 = 0.95$ and $\varepsilon = 0.1$ to (5), we can obtain $n_0 \geqslant 185$. Thus, we set a user's certificate has historical data of ten time-slots. In every time slot, the certificate has twenty legal actions and zero suspicious actions. Consequently, there are two hundred samples in the historical data. The ten historical time slots are numbered from the −9th time slot to the 0th time slot. After the 0th time slot, the certificate will have zero legal actions and twenty suspicious actions in each time slot. T_{C-d} in each time slot will calculate based on the data of all historical time slots.

Our trust model is compared with that in [16–18]. In these previous researches, an exponential function with base μ_1 has been utilized to decay the positive and negative counts in the Bayesian trust system. The exponential function can be expressed as below:

$$\eta = \mu_1^{\Delta t}, \tag{24}$$

where μ_1 is the decay factor of the positive or negative count, $\mu_1 \in [0, 1]$, Δt is the period from the time slot when the positive or negative count recorded to the current time slot. The comparison between using our forgetting curve and the exponential function with base μ to calculate T_{C-d} is presented in Fig. 6.

Fig. 6a shows T_{C-d} calculated by our trust model, in which the decay function is the exponential function with base e, in different time slots. λ_1 is set to 34, 13 and 6 separately. The situation without time decay is also compared. T_{C-d} decreases with time. The decay function makes T_{C-d} declined faster than the situation without time decay. The smaller λ_1 is, the faster the decay rate of T_{C-d} will be.

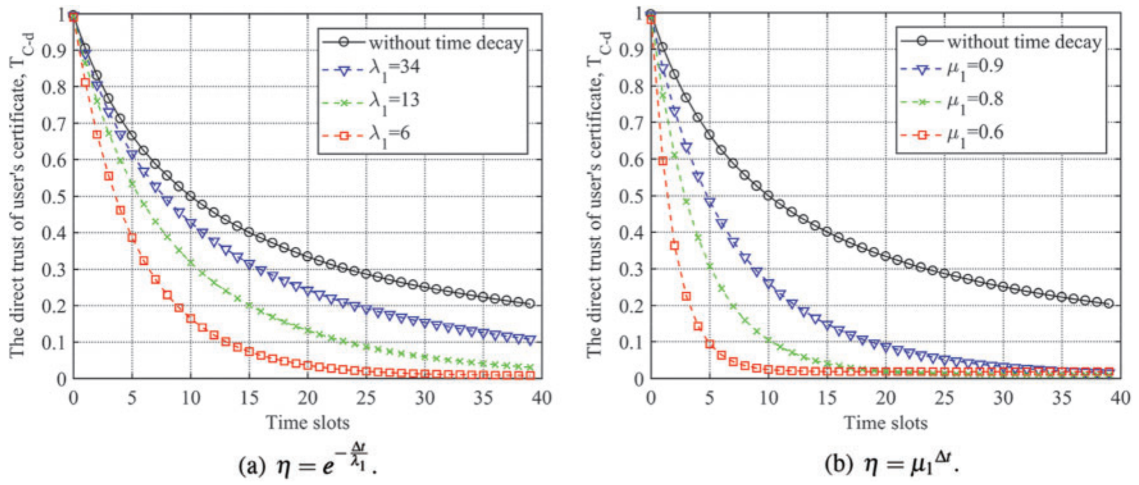


Figure 6: A comparison of using two decay functions to calculate T_{C-d} when λ_1 and μ_1 take different values

In the 10th time slot, the count of legal and suspicious actions are equal, which is two hundred. Without time decay, T_{C-d} is 0.5 in this time slot. The choice of λ_1 depends on how fast T_{C-d} is excepted to decay. For example, we assume that a certificate with suspicious actions no less than half is defined as in the *low* trust level. Thus, T_{C-d} should be less than the upper bound of the *low* trust level beginning from the 10th time slot. If the upper bound of the *low* trust level is 0.43. T_{C-d} is 0.4275 when $\lambda_1 = 34$, which is the maximal integer that makes T_{C-d} less than 0.43 in the 10th time slot. Thus, λ_1 is suggested to be 34.

Fig. 6b illustrates T_{C-d} calculated by the decay function, which is the exponential function with base μ_1, proposed in previous researches in different time slots. μ_1 is set to 0.9, 0.8 and 0.6 separately. The situation without time decay is also compared. T_{C-d} declines with time. The decay function makes T_{C-d} dropped faster than the situation without time decay. The smaller μ_1 is, the faster the decay rate of T_{C-d} will be. When $\mu_1 = 0.9$, T_{C-d} is 0.2613 in the 10th time slot. When $\mu_1 = 0.8$, T_{C-d} is 0.1 in the 10th time slot. Moreover, when $\mu_1 = 0.6$, T_{C-d} attenuated to 0.1 in the 5th time slot. The decay rate of the exponential function with base μ_1 is too fast. A tiny decrease in μ_1 will lead to a huge drop in the decay rate. Taking the same example as in our scheme, if the upper bound of the *low* trust level is 0.43, μ_1 should be greater than 0.9 and less than 1. The choice scope of μ_1 is very narrow. Hence, the decay rate in our scheme is more appropriate.

6.1.2 λ_2

λ_2 is the decay rate factor of the active count of user's or CA's certificate for calculating ω_1 and ω_2. We choose the active days in the recent six months to calculate the user activity. One time slot is set as a month. The maximal active count of a time slot is 31 days. The maximal count of user activity is defined as 30 active days in each of the three months and 31 active days in each of the other three months. ω_1 for the maximal count of user activity is calculated for selecting λ_2. The half-life of the active count, t_0, is also calculated for comparison. According to (21), t_0 can be calculated as follows:

$$t_0 = \lambda_2 \times \ln 2. \tag{25}$$

The forgetting curve in our trust model is also compared with an exponential function with base μ_2. The exponential function is modeled as follows:

$$\varphi = {\mu_2}^{\Delta t}, \tag{26}$$

where μ_2 is the decay factor of the active count of the user's or CA's certificate, $\mu_2 \in [0, 1]$, Δt is the period from the time slot when the active count recorded to the current time slot. ω_1 of the certificate with the maximal count of user activity and t_0 when λ_2 and μ_2 take different values are depicted in Fig. 7.

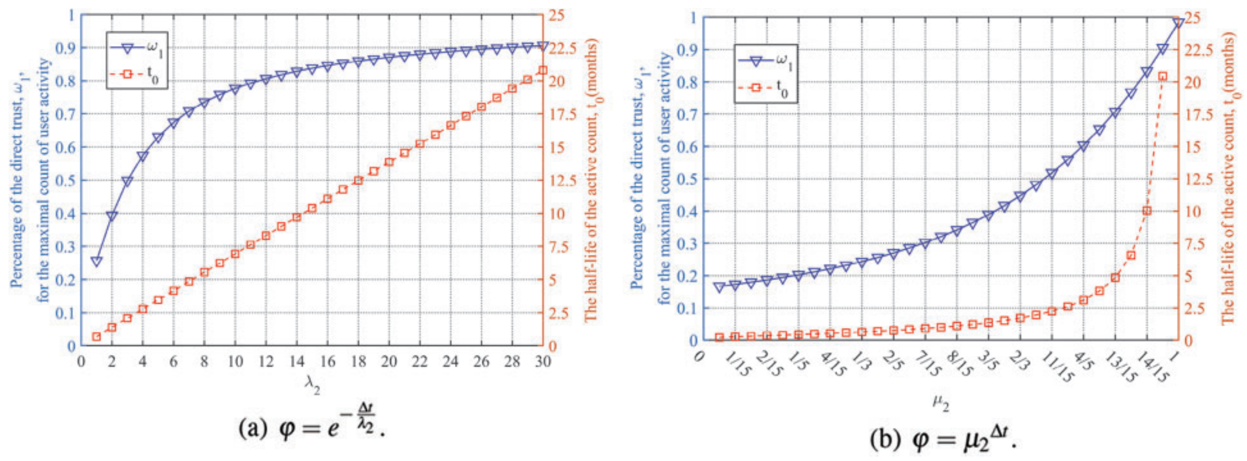


Figure 7: A comparison of using two decay functions to calculate ω_1 and t_0 when λ_2 and μ_2 take different values

Fig. 7a demonstrates ω_1 for the maximal count of user activity and t_0 calculated by our trust model when λ_2 takes different values. The exponential function with base e is utilized as the decay function. ω_1 for the maximal count of user activity and t_0 both raise when λ_2 increases. The rising rate of ω_1 is slower and slower, while t_0 has linear growth.

The enhancement of ω_1 is at the cost of the extension of t_0. t_0 should not be far greater than the period used to calculate user activity, which is six months in this experiment. If t_0 is too big, the attenuated effect of the decay function for the active count is too faint. Meanwhile, ω_1 for the maximal count of user activity should be as large as possible. Therefore, it should make a balance between ω_1 and t_0. When λ_2 enhances from 12 to 30, t_0 increases 12.5 (from 8.3 to 20.8); meanwhile, ω_1 rises just 0.1 (from 0.8 to 0.9). The rising rate of ω_1 is too slow, and the cost of t_0 is too expensive when ω_1 is greater than 0.8. Thus, λ_2 is suggested to be 12 as ω_1 for the maximal count of user activity is 0.8, and t_0 is 8.3.

Fig. 7b presents ω_1 for the maximal count of user activity and t_0 calculated by the decay function proposed in previous researches. The exponential function with base μ_2 is applied as the decay function. t_0 and ω_1 for the maximal count of user activity both increase as μ_2 grows, and the rising rates both are faster and faster. When $\mu_2 = 0.9$ (the point between 13/15 and 14/15), ω_1 for the maximal count of user activity is 0.77, and t_0 is 6.58. As mentioned above, t_0 should be a bit more than six, and ω_1 for the maximal count of user activity should be as large as possible. Thus, μ_2 should be larger than 0.9. However, the growth rate of t_0 and ω_1 is too fast when μ_2 is greater than 0.9. Therefore, a tiny growth in μ_2 will lead to a huge rise in t_0 and ω_1. For example, when $\mu_2 = 14/15 \approx 0.933$, ω_1 for the maximal count of user activity is 0.83, and t_0 is 10.05. This value of t_0 is too large as the period for calculating user activity is six months. Thus, the appropriate value of μ_2 is in (0.9, 0.933). The scope for choosing μ_2 is also very narrow. Hence, it is harder to find an appropriate value of μ_2 to balance the value of t_0 and ω_1 compared with our scheme.

6.2 *The Accuracy of Detection*

Our link congestion detection utilizes the method proposed in [6]. The detection accuracy of link congestion is 100%, as shown in [6]. Moreover, the Crossfire attack identification depends on the judgment of the connectivity of the graph constructed by uncongested links. The connectivity of a given graph is determinate. Therefore, the accuracy of the Crossfire attack identification is also 100%.

6.3 *The Effectiveness of Congestion Mitigation*

Assuming a trust model has been given, we illustrate how the distribution of the certificates in each trust level affects the link congestion mitigation in this section. We evaluate two contrasting distribution of the certificates. The two distributions are shown as follows:

1) The number of certificates in each trust level is imbalanced, denoted as $distribution_1$.
2) The number of certificates in each trust level is equal, denoted as $distribution_2$.

In $distribution_1$, the most common case is that the certificates are concentrated at the *medium* level. Thus, we choose the normal distribution as an example. We assume the trust values of the certificates obey the normal distribution $N(1/2, (1/6)^2)$. In a normal distribution, about 68.2% of values are within one standard deviation away from the mean; about 95.5% of the values lie within two standard deviations; about 99.7% are within three standard deviations. Therefore, 99.7% of the trust value are within (0, 1) in $N(1/2, (1/6)^2)$. Dividing (0, 1) into three intervals equally, about 68.2% of the certificates are in the *medium* trust level. And the *high* and *low* trust level each account for about 15.9%. Inputting the following values (27) into Algorithm 2, we can obtain the bandwidth threshold for the queues of different trust levels in $distribution_1$, as shown in Table 1.

$$\begin{cases} BW = 5\ Mbps, \\ NUM_h = NUM_l = 15.9\%, \\ NUM_m = 68.2\%, \end{cases} \tag{27}$$

Table 1: Bandwidth threshold for the queues of different trust levels in $distribution_1$

The trust level of the queue	Max-rate	Min-rate
High	5 Mbps	2.76 Mbps
Medium	4.21 Mbps	2.24 Mbps
Low	0.8 Mbps	0

In $distribution_2$, about 33.33% of the certificates are in each trust level. Thus, the bandwidth threshold for the queues of different trust levels in $distribution_2$, as depicted in Table 2, can be derived by taking the following values (28) into Algorithm 2 as input.

$$\begin{cases} BW = 5\ Mbps, \\ Num_h = Num_m = Num_l = 33.33\%, \end{cases} \tag{28}$$

We use Mininet to simulate an SDN system. Mininet is run on a laptop with Intel Core i7-7600U 2.8 GHz CPU and 16 GB memory. A Floodlight OpenFlow controller is run on the same laptop. The network topology in Mininet is shown in Fig. 8. The seven OpenFlow switches are all managed by the Floodlight controller. The traffic from three trust levels is sent from the IoT devices *Host1*, *Host2* and *Host3*, and received by the decoy servers *Decoy1*, *Decoy2* and *Decoy3*. The traffic is generated and received by the software D-ITG [33]. The target links are the link between the SDN switch *s1* and *s7* and between *s7* and *s2*. The target switch is *s7*. The network behind *s7* is the target area. The bypass switch *s6* is only used for rerouting in the comparison experiment. The bandwidth of the target link and rerouting link is both five megabits per second (Mbps).

Table 2: Bandwidth threshold for the queues of different trust levels in $distribution_2$

The trust level of the queue	Max-rate	Min-rate
High	5 Mbps	3.33 Mbps
Medium	3.33 Mbps	1.67 Mbps
Low	1.67 Mbps	0

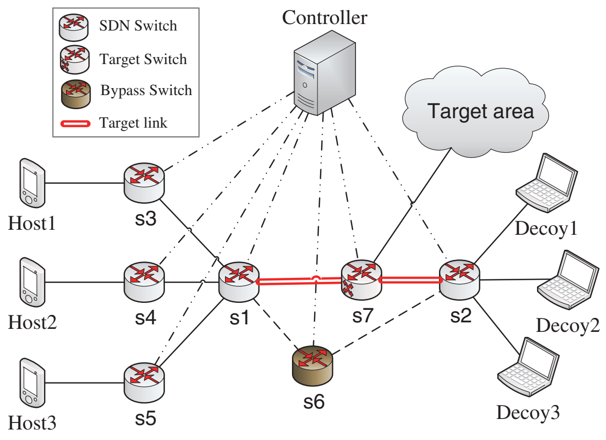
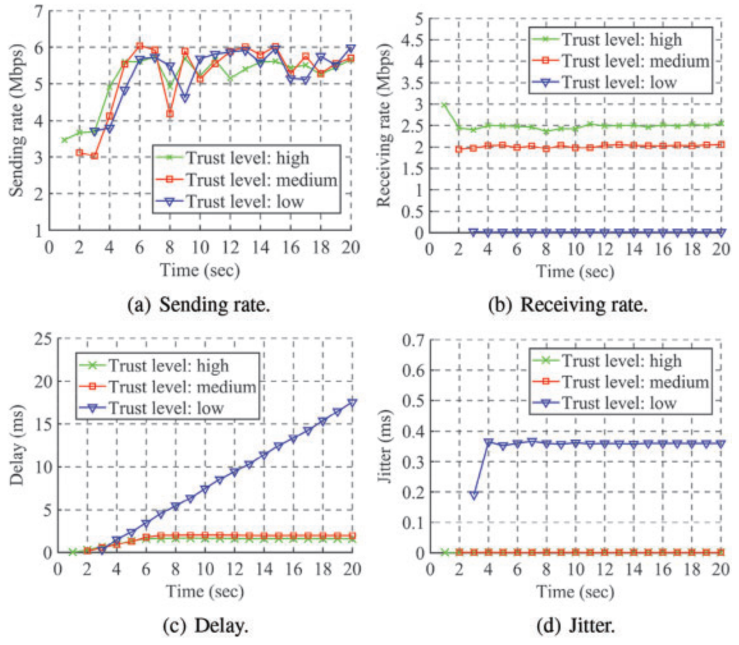


Figure 8: The network topology in the simulation

Our framework is compared with the rerouting mechanism proposed in [6,34]. Optimized Packet Distributor (OPD) [34] is the most recent SDN-based rerouting scheme for mitigating the Crossfire attack. OPD uses the same attack detection mechanism as [6]. After the attack has been detected, the traffic is rerouted by a load balance algorithm. However, OPD has not identified the malicious traffic and just mitigated the link congestion. To evaluate our scheme in the worst condition, we ignore the run time of the load balance algorithm in [34]. Thus, Al Farooq et al. [34] have the same rerouting performance as [6].

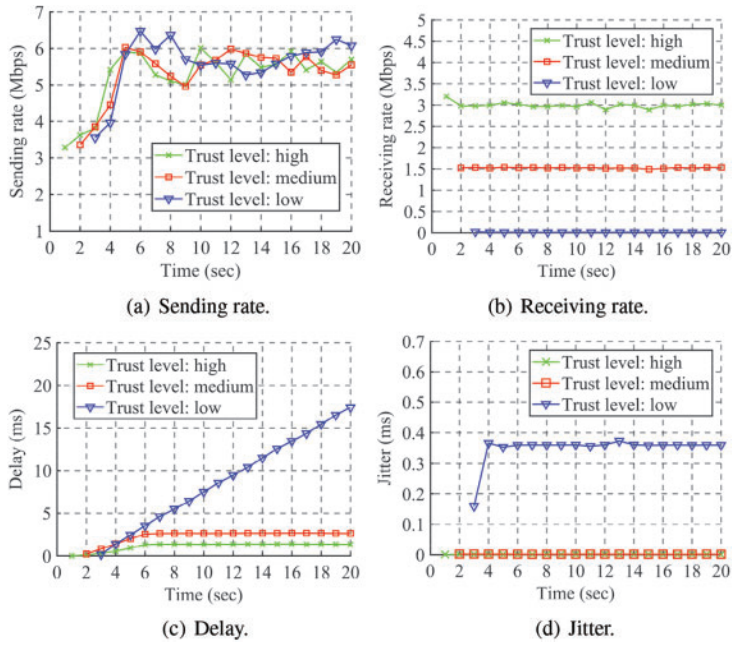
We evaluate the effectiveness of differentiated services through the following parameters: throughput, delay and jitter. The performance evaluation of $distribution_1$ and $distribution_2$ is presented in Figs. 9 and 10, respectively. Figs. 9a and 10a show the sending rates of the traffic from the three trust levels over time in the two distributions, respectively. The settings of traffic sending in the two distributions of the certificates are the same. After the 5th second, the sending rates of the traffic from the three trust levels are around 5 Mbps, which is the bandwidth of the target link, to ensure the target link is congested. The traffic from each trust level lasts for 30 s. The data in the first 20 s are presented. Moreover, the traffic from the three trust levels is sent successively at intervals of one second in the beginning. The traffic from the *high* trust level began first, then the *medium* trust level and the *low* trust level last.



(a) Sending rate. (b) Receiving rate.

(c) Delay. (d) Jitter.

Figure 9: The performance evaluation of *distribution*$_1$



(a) Sending rate. (b) Receiving rate.

(c) Delay. (d) Jitter.

Figure 10: The performance evaluation of *distribution*$_2$

The receiving rates of the traffic from the three trust levels over time in the two distributions are illustrated in Figs. 9b and 10b, respectively. In Fig. 9b, the receiving rate of the traffic from the *high* trust level is around 2.5 Mbps, from the *medium* trust level is around 2 Mbps, from the *low* trust level is almost zero. In Fig. 10b, the receiving rate of the traffic from the *high* trust level is around 3 Mbps, from the *medium* trust level is around 1.5 Mbps, from the *low* trust level is almost zero. The receiving rate of the traffic from every trust level is restrained below the corresponding *min-rate* shown in Tables 1 and 2.

The congestion on the traffic from the *high* trust level obtains the most mitigation. In contrast, the traffic from the *low* trust level receives the most restraint. Meanwhile, relatively fair routing services are provided to the traffic from the *medium* trust level. The number of the *medium* trust level certificates in $distribution_1$ (account for 68.2%) is more than that in $distribution_2$ (account for 33.33%). The receiving rate of the traffic from the *medium* trust level in Fig. 9b is higher than that in Fig. 10b. In other words, as the ratio of certificates in the *high* trust level increases, the restraining of the traffic from the other two trust levels will also enhance.

Figs. 9c and 10c depict the delay of the traffic from the three trust levels over time in the two distributions, respectively. The delay of the traffic from the *low* trust level has a linear growth in both distributions. After the 5th second, the delay of the traffic from the *high* and *medium* trust levels becomes nearly constant at a small value. In Fig. 9c, the delay of the traffic from the *high* trust level is about 1.6 ms, from the *medium* trust level is about 2.0 ms after the 5th second. In Fig. 10c, the delay of the traffic from the *high* trust level is about 1.3 ms, from the *medium* trust level is about 2.6 ms after the 5th second. The linear growth of the delay represents that the traffic from the *low* trust level is restrained, and the transmitting time keeps increasing. The delay of the traffic from the *medium* trust level is greater than that from the *high* trust level in both distributions, as the latter has a higher receiving rate. The delay of the traffic from the *high* trust level in Fig. 9c is more than that in Fig. 10c for the same reason. Moreover, the delay of the traffic from the *medium* trust level in Fig. 9c is smaller than that in Fig. 10c. This difference is because the receiving rate of the traffic from the *medium* trust level in Fig. 9b is greater than that in Fig. 10b.

The jitter of the traffic from the three trust levels over time in the two distributions is presented in Figs. 9d and 10d, respectively. The jitter of the traffic from the *high* and *medium* trust level is almost zero in both distributions. In contrast, the jitter of the traffic from the *low* trust level is much more significant as nearly 0.4 ms in both distributions. The almost zero jitter means that the traffic from the *high* and *medium* trust level is received with a stable bandwidth. In contrast, a significant jitter indicates that the traffic from the *low* trust level is restrained to almost zero and received intermittently.

The evaluation of the rerouting scheme is portrayed in Fig. 11 for comparison. The sending rates, receiving rates, delay and jitter of the traffic from three trust levels over time are shown in Figs. 11a–11d, respectively. The sending rate of each trust level is around 5 Mbps. The traffic sending lasts for 60 s. We select the data in 20 s around the point rerouting happened for presenting in Fig. 11. Half of the traffic on the SDN switch *s1* is rerouted to the SDN switch *s6* at the 11th second. The receiving rates of the traffic from the three trust levels are around 1.5 Mbps before rerouting and around 3 Mbps after rerouting, as illustrated in Fig. 11b. Fig. 11c depicts the delay of the traffic from the three trust levels all descends when the rerouting happens. The delay goes up to the value before rerouting in several seconds because the overall bandwidth is still less than the total sending rate after rerouting. The jitter of the traffic from the three trust levels is also enhanced when the rerouting happens and then goes down to nearly zero in several seconds, as revealed in Fig. 11d. The traffic from the three trust levels has the same performance in the rerouting scheme.

Comparing with the rerouting scheme, Certrust can keep a high quality of services, such as higher throughput, lower delay and lower jitter, for the traffic from the *high* trust level from the beginning of link congestion. In the rerouting scheme, the congestion is required to be detected firstly and then change the routing policy in the network. The service quality will not change until the rerouting works. Moreover, the traffic from any trust levels is treated equally in the rerouting scheme.

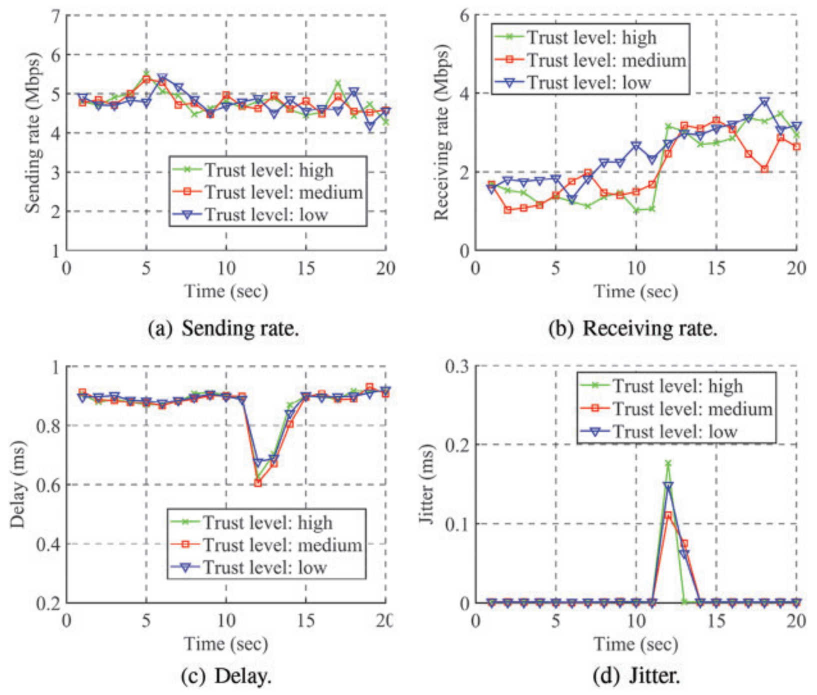


Figure 11: The evaluation of the rerouting scheme

6.4 Computational Overhead of Controller

The computational overhead of the controller for certificate verification has been evaluated in DTLShps [15]. In DTLShps, the computational overhead of the controller is evaluated under the number of concurrent connections from 1 to 30. To evaluate our scheme in the worst condition, we use the data when the number of concurrent connections is 30. When the number of concurrent connections is 30, DTLShps increases the overhead by about 1.7 times compared with the traditional DTLS in SDN. Our framework adds a step of the trust level query compared with the handshake process in DTLShps. The computational overhead of the trust level query for a certificate depends on the search algorithm chosen and the size of the record. For example, the time complexity of *Linear Search* is $O(n)$ [35], *Jump Search* is $O(\sqrt{n})$ [36], and *Binary Search* is $O(log_2\ n)$ [35]. The trust level of certificates and the count of network behaviors are calculated in the background. Thus, these calculating processes do not influence the delay in the handshake phase.

In our experimental environment, the average computational overhead on the controller for the traditional DTLS handshake is 155 ms. The scheme of certificate verification increases this overhead by 1.7 times, as illustrated in [15]. Thus, the computational overhead on the controller for the scheme of certificate verification is $155 \times (1 + 1.7) = 418.5$ ms. The average time for searching the trust level of a certificate in a record with 100 items using the linear search algorithm is far less than 1 ms in the worst condition and can be ignored. The average time from the controller sent the rerouting policy to the delay of the traffic started falling, as shown in Fig. 11c, is in the range of 1 to 2 s. As the statistical frequency of our experiment is per second, we can not obtain a more precise value of the extra delay for rerouting. All the average values above are calculated over 100 samples. Although inevitably increasing the computational overhead on the controller in the handshake phase, Certrust has no delay in mitigating the link congestion. In contrast, the rerouting scheme adds no delay in the handshake phase but inevitably generates extra delay for rerouting.

7 Conclusion

We proposed a certificate-based framework, Certrust, to mitigate the Crossfire attack over SDN in the IoT scenarios. The malicious flows in the Crossfire attack were identified by the trust level of the certificates. The trust of the certificate was modeled by the Bayesian trust system combining with the forgetting curve. Moreover, the trust model was calculated based on the participation of the certificate in each Crossfire attack. For detecting the attack accurately, a method based on graph connectivity was also proposed to detect the Crossfire attack from the link congestion events. Finally, four trust-based routing principles and their implementation were proposed to mitigate the link congestion caused by the Crossfire attack.

Certrust settles link congestion in real-time and identifies the Crossfire attack in the background. The link congestion is mitigated at the beginning without any delay for attack detection. The user with a good reputation will receive a high-quality service under the Crossfire attack. In contrast, the user with a poor reputation will suffer a hard time under the Crossfire attack. The traffic from users, who do not use certificates, is treated the same as the traffic from certificates with a medium reputation. If the attack traffic does not use certificates, the attack traffic will also be restrained compared with the traffic from certificates with a good reputation. Thus, the service quality of the traffic from certificates with a good reputation will always be guaranteed. This guarantee will also encourage users to use certificates in communication.

The experiment illustrated that the decay rate of our forgetting curve is more moderate and appropriate for the Bayesian trust system than that of the decay function proposed in the previous researches. Although the delay in the handshake phase of a secure session is inevitably increased by about 1.7 times, Certrust performed more effectively in mitigating link congestion than the rerouting schemes.

With the rapid development of IoT, certificates will be more and more widely used. Although certificates bring extra overhead compared with Pre-Shared Key (PSK) for authentication in IoT devices, one certificate can be authenticated by multiple communication peers. On the contrary, one PSK can correspond only to one communication peer, and the PSK also needs to be configured on the communication peer previously. The more peers an IoT device needs to communicate with, the more prominent the advantage of certificates will be. Furthermore, it is more and more popular to disguise malicious traffic as regular traffic to improve the success rate of attacks. Thus, with the wide usage of certificates in the future, more and more attacks will also use certificates. Therefore, the proposed trust model of certificates will be applied to protect against more attacks in the future.

Funding Statement: This work was supported by Joint Funds of the National Natural Science Foundation of China and Xinjiang under Project U1603261.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Kolias, C., Kambourakis, G., Stavrou, A., Voas, J. (2017). DDoS in the IoT: Mirai and other botnets. *Computer, 50(7),* 80–84. DOI 10.1109/MC.2017.201.
2. Ubale, T., Jain, A. K. (2020). Survey on DDoS attack techniques and solutions in software-Defined network. In: Gupta, B. B., Perez, G. M., Agrawal, D. P., Gupta, D. (Eds.), *Handbook of computer networks and cyber security: Principles and paradigms*, pp. 389–419. Cham: Springer International Publishing.
3. Kang, M. S., Lee, S. B., Gligor, V. D. (2013). The crossfire attack 2013. *IEEE Symposium on Security and Privacy*, Berkeley, CA, USA.
4. Aydeger, A., Saputro, N., Akkaya, K. (2019). A moving target defense and network forensics framework for ISP networks using SDN and NFV. *Future Generation Computer Systems, 94,* 496–509. DOI 10.1016/j.future.2018.11.045.
5. Xie, L., Ding, Y., Yang, H., Hu, Z. (2020). Mitigating LFA through segment rerouting in IoT environment with traceroute flow abnormality detection. *Journal of Network and Computer Applications, 164,* 102690. DOI 10.1016/j.jnca.2020.102690.
6. Rafique, W., He, X., Liu, Z., Sun, Y., Dou, W. (2019). CFADefense: A security solution to detect and mitigate crossfire attacks in software-defined IoT-edge infrastructure. *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, Zhangjiajie, China.
7. Wang, J., Wen, R., Li, J., Yan, F., Zhao, B. et al. (2019). Detecting and mitigating target link-flooding attacks using SDN. *IEEE Transactions on Dependable and Secure Computing, 16(6),* 944–956. DOI 10.1109/TDSC.8858.
8. Narayanadoss, A. R., Truong-Huu, T., Mohan, P. M., Gurusamy, M. (2019). Crossfire attack detection using deep learning in software defined ITS networks. *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, Kuala Lumpur, Malaysia.
9. Narten, T., Draves, R., Krishnan, S. (2007). Privacy extensions for stateless address autoconfiguration in IPv6 (No. rfc4941).
10. Xing, J., Yang, M., Zhou, H., Wu, C., Ruan, W. (2019). Hiding and trapping: A deceptive approach for defending against network reconnaissance with software-defined network. *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*, London, UK.
11. Perkins, C. (2010). IP mobility support for IPv4, revised (No. rfc5944).
12. Perkins, C., Johnson, D., Arkko, J. (2011). Mobility support in IPv6 (No. rfc6275).
13. Hamad, S. A., Sheng, Q. Z., Zhang, W. E., Nepal, S. (2020). Realizing an internet of secure things: A survey on issues and enabling technologies. *IEEE Communications Surveys Tutorials, 22(2),* 1372–1391.
14. Firefox (2020). Percentage of web pages loaded by firefox using HTTPS. https://letsencrypt.org/stats/.
15. Ma, Y., Yan, L., Huang, X., Ma, M., Li, D. (2020). DTLShps: SDN-based DTLS handshake protocol simplification for IoT. *IEEE Internet of Things Journal, 7(4),* 3349–3362.
16. Jiang, F., Tseng, H. W. (2019). Trust model for wireless network security based on the edge computing. *Microsystem Technologies, 27(4),* 1627–1632.
17. Kanchana Devi, V., Ganesan, R. (2019). Trust-based selfish node detection mechanism using beta distribution in wireless sensor network. *Journal of ICT Research and Applications, 13(1),* 79–92.
18. Singh, P., Gill, N. S. (2020). Self-observation and recommendation based trust model with defense scheme for wireless ad hoc network. *Journal of High Speed Networks, 26(1),* 41–54.
19. Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L. et al. (2013). B4: Experience with a globally-deployed software defined WAN. *ACM SIGCOMM Computer Communication Review, 43(4),* 3–14.
20. Wu, X., Huang, J., Ling, J., Shu, L. (2019). BLTM: Beta and LQI based trust model for wireless sensor networks. *IEEE Access, 7,* 43679–43690.

21. JÃşang, A., Ismail, R. (2002). The beta reputation system. *Proceedings of the 15th Bled Electronic Commerce Conference*, Bled, Slovenia.
22. Mui, L., Mohtashemi, M., Halberstadt, A. (2002). A computational model of trust and reputation. *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, Big Island, HI, USA.
23. Wang, W., Zeng, G. S. (2007). Trusted dynamic level scheduling based on Bayes trust model. *Science in China Series F: Information Sciences, 50(3),* 456–469.
24. Park, J., Kwon, H., Kang, N. (2017). IoT–cloud collaboration to establish a secure connection for lightweight devices. *Wireless Networks, 23(3),* 681–692. DOI 10.1007/s11276-015-1182-y.
25. Park, J., Kang, N. (2014). Lightweight secure communication for CoAP-enabled internet of things using delegated DTLS handshake. *2014 International Conference on Information and Communication Technology Convergence (ICTC)*, Busan, Korea (South).
26. Hummen, R., Shafagh, H., Raza, S., Voig, T., Wehrle, K. (2014). Delegation-based authentication and authorization for the IP-based Internet of Things. *2014 Eleventh Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, Singapore.
27. Park, S., Park, H., Lee, J., Kent, S. (2009). Traceable anonymous certificate. *Request for Comments (RFC)*, 5636.
28. Liu, H., Cui, L., Yao, T., Li, R. (2019). A personalized recommendation method based on collaborative filtering algorithm. *International Journal of Business and Economics Research, 8(5),* 297–302. DOI 10.11648/j.ijber.20190805.16.
29. Averell, L., Heathcote, A. (2011). The form of the forgetting curve and the fate of memories. *Journal of Mathematical Psychology, 55(1),* 25–35. DOI 10.1016/j.jmp.2010.08.009.
30. Kong, W., Li, X., Hou, L., Li, Y. (2020). An efficient and credible multi-source trust fusion mechanism based on time decay for edge computing. *Electronics, 9,* 502. DOI 10.3390/electronics9030502.
31. Studer, A., Perrig, A. (2009). The coremelt attack. In: Backes, M., Ning, P. (Eds.), *Computer security–ESORICS 2009*, Berlin, Heidelberg: Springer Berlin Heidelberg.
32. Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. (2009). *Introduction to algorithms* (3rd edition). Cambridge, MA, USA: MIT Press.
33. Botta, A., Dainotti, A., Pescapè, A. (2012). A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks, 56(15),* 3531–3547. DOI 10.1016/j.comnet.2012.02.019.
34. Al Farooq, A., Moyer, T., Ahmed, D. T. (2021). OPD: Network packet distribution after achieving equilibrium to mitigate DDOS attack. *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, Madrid, Spain.
35. Parmar, P. V., Kumbharana, C. (2015). Comparing linear search and binary search algorithmsto search an element from a linear list implementedthrough static array, dynamic array and linked list. *International Journal of Computer Applications, 121(3),* 13–17.
36. Shneiderman, B. (1978). Jump searching: A fast sequential search technique. *Communications of the ACM, 21(10),* 831–834. DOI 10.1145/359619.359623.