UC Santa Cruz

UC Santa Cruz Previously Published Works

Title

Secure Packet Forwarding in a Computer Network or VPN Using Cryptographic Entanglements

Permalink

https://escholarship.org/uc/item/7f75b4ds

Journal

IEEE IPCCC 2026

Authors

Garcia-Luna-Aceves, J.J.

Moghaddassian, Morteza

Publication Date

2026-11-20

Data Availability

The data associated with this publication are within the manuscript.

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

Secure Packet Forwarding in a Computer Network or VPN Using Cryptographic Entanglements

J.J. Garcia-Luna-Aceves and Morteza Moghaddassian
Electrical and Computer Engineering Department
University of Toronto
Ontario, Canada (M5S 3G4)
Email: jj.garcialunaaceves@utoronto.ca, m.moghaddassian@utoronto.ca

***Abstract*—The headers of IP packets are transmitted in the clear and are therefore open to many types of attacks. Secure packet forwarding can be accomplished today by means of Virtual Private Networks (VPN) based on IPsec. However, IPsec has a number of limitations that stem from its complexity, its reliance on public keys, and the need to use a key-exchange protocol. We introduce the use of cryptographic entanglements as an IPsec alternative to enable secure packet forwarding within a computer network or a VPN managed by the same authority. Trusted nodes are configured with the same two secret integers shared by all trusted routers of the computer network or VPN, and any two trusted nodes use that limited information to become cryptographically entangled in the use of encryption keys. Such keys are used to hide the true identities of the sources and destinations of IP datagrams as well as the payload carried in IP datagrams. It is shown that the new approach provides strong security against known attacks, and performance results of a prototype implementation of cryptographic entanglements show that they are a viable approach to secure the forwarding plane of a computer network or establish a VPN.**

I. Introduction

Standard packet forwarding in a computer network using the Internet Protocol (IP) is based on packet headers sent in plain text. Unfortunately, using plain-text source and destination addresses open a computer network to a variety of attacks, such as the forging of IP addresses, eavesdropping and traffic analysis, main-in-the-middle attacks, and even the tampering of routing tables. A number of approaches have been proposed over the years to detect or track attacks or reduce packet-forwarding vulnerabilities in computer networks [4]. The approaches intended to reduce packet-forwarding vulnerabilities have relied on changes to the basic hop-by-hop forwarding of datagrams and some also required changes to the control plane. Examples of these changes include using filters at destination routers, using source routes (e.g., [11]), using path traces in packet (e.g., [9], [10], or changing forwarding tables (e.g., [4]).

Given the vulnerable nature of standard IP packet forwarding in computer networks, Virtual Private Networks (VPN) have been used to support secure and private packet forwarding [1]. The standard method to implement a VPN spanning multiple routers today is based on IPsec [3].

Confidentiality, authentication and integrity for packet forwarding can be provided with IPsec based on the Encapsulating Security Payload (ESP) protocol [8]. However, the approach is complex and relies on using handshakes for key exchanges [7] a public key infrastructure, which makes it vulnerable to attacks that stem from such factors as configuration complexity, misconfigured gateways, and using pre-shared keys [2].

This paper uses the concept of *cryptographic entanglement* introduced in [5] to eliminate the need for a key-distribution method and as the basis to establish secure packet forwarding in a computer network or VPN managed by the same authority. A cryptographic entanglement consists of two or more nodes reaching a state of correlated processing of information by first sharing a few secrets different than the encryption keys they use in the entanglement. Once nodes reach and maintain a cryptographic entanglement, they derive the same sequences of encryption keys to encrypt and decrypt messages without requiring messages to be sent reliably and without ever having to communicate the keys themselves. We focus on implementing cryptographic entanglements by configuring in advance all trusted nodes with the same few secrets through some secure means, and we address the use of cryptographic entanglements to secure a computer network or Virtual Private Network (VPN) administered by a single authority.

Section II describes CONCISE (Cryptosystem for Obfuscation, Non-repudiation and Confidentiality with Integer Set Entanglements) [5], which is an approach to implement cryptographic entanglements. Trusted nodes in a network using CONCISE are configured with the same *Baseline for Entanglement Trust* (BET) and the *jump base*. A sender and one or multiple receivers that share the same BET and jump base become *cryptographically entangled* as soon as the sender node starts sending encrypted messages to the receiver node using symmetric encryption keys derived from their shared secrets using common functions. We focus on XOR-based ciphers because of their simplicity.

To allow the sender and receivers to maintain their correlated selection of encryption keys, the ciphertext sent consists of a header and a payload. The header states the metadata needed for the receivers to use the correct encryption keys even if messages are lost or arrive out of order, but without providing any insight on the encryption keys themselves or the structure of the payload. Importantly, this is attained without the need for any handshake that may divulge any secrets, and without the need to explicitly configure trusted nodes with encryption keys.

Section III describes how CONCISE can be used to enable secure packet forwarding in a computer network managed by the same authority, and how it can be used as an alternative to IPsec to establish a VPN managed by the same authority. In a companion paper [6], we describe how cryptographic entanglements can be also used to secure the signalling messages of routing protocols operating in the control plane.

Section IV shows that CONCISE is secure against known attacks, and Section V discusses experimental results showing the performance of XOR ciphers using encryption keys from entanglements. The experiment assume messages being sent in IPv4 packets with the payload of each packet containing the CONCISE header and the ciphertext payload. The results show that the average processing rates attained are above 2.4 Gbps, which indicates that CONCISE can be used to secure the forwarding plane of a computer network or VPN.

II. Cryptographic Entanglements in CONCISE

A. Cryptographic Entanglements

In this paper we assume that all trusted nodes in a network using CONCISE are configured with the same secrets as pre-shared knowledge known only to them. These secrets are the *Baseline for Entanglement Trust* (BET) and the *jump base* (J).

The configured value of the BET is denoted by $\mathcal{B}$, and it is a positive integer of length $L_{\mathcal{B}}$ such that $\mathcal{B} > 2^{L_{\mathcal{B}}-1}$. In practice, $L_{\mathcal{B}}$ need not depend on the required lengths of encryption keys used in any of the ciphers that can be adopted in CONCISE. The value of J is a relatively small integer (e.g., $2 \leq J \leq 100$).

The sender and receivers of an entanglement use a common *anchor function* based on $\mathcal{B}$, J, and the identifier of the sender in the entanglement to agree on another integer called the *entanglement anchor* and denoted by E. The resulting value of E is such that $E > 2^{L-1}$, where $L > L_{\mathcal{B}}$ is the initial length of integers used to derive encryption keys in an entanglement.

Let $\mathcal{L}_q[n]$ denote the integer corresponding to the q least significant bits (LSBs) of the binary representation of a positive integer n. An example of an anchor function to compute E is

$$E = \mathcal{L}_L[(\mathcal{B} + a) \cdot 2^L + J] \tag{1}$$

The entanglement anchor computed by the sender and receivers defines the integer set $\mathcal{I}$ used in the entanglement

$$\mathcal{I} = \{\ n \in Z^+ \mid E \leq n \leq 2^L - 1\ \} \tag{2}$$

Depending on the cipher used in CONCISE, the payload of a CONCISE message may be organized into one or multiple message blocks. If message blocks are used in the cipher, each block m_i is of length L and padding is used in the last chunk of the message if needed.

We denote by k_0 the *basis key* from set $\mathcal{I}$ computed for each message. The basis key is used to derive one or multiple encryption keys of a given length needed by the specific cipher used in CONCISE. If the cipher requires one encryption key per message block, the key used for ith message block is denoted by k_i, where $i \geq 0$. If a single encryption key is used by the cipher, that encryption key is denoted by k_1.

B. Relative Anchors and Relative Indices

To cope with packet losses, each message must give the receiver a hint about the basis key k_0 that was used, without revealing its actual value of the keys or divulging E. This hint is based on a relative index in set $\mathcal{I}$. Because set $\mathcal{I}$ may be very large, it is organized into *entanglement groups*, with each group containing 2^R integers and $R \ll L$ to enable the use of relative indices of much smaller length than L. For example, R could be just 32 bits while L could be thousands of bits. The first entanglement group starts with E.

A *relative anchor* is defined to be the integer at the start of an entanglement group, and it is used as the point of reference for the *relative index* stated in a message header in place of a key value. The relative anchor assumed by the sender of an entanglement (node a) in its latest message to node b is denoted by $A(a)$.

The relative index that node a uses in the header of a message to node b instead of a key value is denoted by $R(a)$. Node a maintains the value of the last encryption key used for the last message sent to node b (denoted by ω).

C. Jumping to a New Relative Anchor

The value of E depends on the identifier of the sender, which is public knowledge. To erase any dependency, on public knowledge node a executes a "jump" on the value of the relative anchor used to determine relative indices stated in message headers.

Given that E can be computed at the receiver based on local information, a jump can be done with the very first message that node a sends to node b. Once a jump to a relative anchor occur, no node other than a and b can easily decipher messages sent from a to b.

The value of a new relative anchor is computed using a function based on J and a random value selected by node a for which node b is given only a hint that depends on J.

Let $A(a, J)$ denote the relative anchor used by node a just before it proceeds with a jump of its relative anchor. The following is an example of a jump function:

$$A(a) = A(a, J) + 2^R(I \cdot 2^I)J \tag{3}$$

The variable I in Eq. (3) is called the *indicator* of the jump. Node a selects the value of I at random from a small range of values (e.g., $0 \leq I \leq 15$). The indicator is used as an exponent and in combination with J to keep the magnitude of the jump a secret and to reduce the header overhead incurred in giving the receiver the hint about the magnitude of the jump by communicating only the indicator value. If no jump takes place, then $I = 0$.

Node a stores the value of I used to determine the last jump of relative anchors in an indicator record $IR(a)$, and this value is used in message headers.

D. Selecting Encryption Keys for New Messages

The length of integers in $\mathcal{I}$ is large enough to accommodate any of the ciphers that can be used in a deployment of CONCISE. The encryption keys used in a message are obtained

from a basis key in set $\mathcal{I}$ and the length of the keys are given by the specific cipher being used.

A sender node a selects a new basis key k_0 from set $\mathcal{I}$ for each new message it sends in an entanglement using a function based on J and a random selection made in such a way that k_0 is larger than the last basis key used.

Let $v \in [1,\ m]_U$ denote an integer v chosen at random from the interval of equally likely integers $[1,\ m]$; an example of a function that node can use a to select a basis key k_0 is given in Eq. (4).

$$k_0 = \begin{cases} A(a) + R_K(a) + u & \text{if } \ IR(a) = 0 \\ A(a) + u & \text{if } \ IR(a) > 0 \end{cases} \qquad (4)$$

$$\text{with } u \in [\ 1,\ 1 + J\mathcal{L}_8[\omega]\ ]_U$$

To obfuscate the sequence of encryption keys used in the same message, the key-selection function uses parameter values that are only known by nodes participating in an entanglement.

Let L_c denote the length of the encryption keys required by the cipher used in CONCISE, a simple example of a key-selection function is the following:

$$k_1 = \mathcal{L}_{L_c}[k_0] \qquad (5)$$

$$k_{i+1} = \mathcal{L}_{L_c}[1 + k_i + \mathcal{L}_4[J] + \mathcal{L}_4[A(a)]] \text{ with } i \geq 1$$

E. Correlating Keys and Jumps without Reliable Handshakes

CONCISE does not do any retransmissions and does not use any handshakes for senders and receivers to use the same encryption keys in the messages they exchange. Instead of reliable handshakes, CONCISE uses a persistence approach to ensure that both ends of an entanglement agree on a new relative anchor after either the sender executes a jump of relative anchors or the value of k_0 is larger than $A(a) + 2^R$.

The approach consists of the receiver using the relative anchor in place before the jump or the needed change of relative anchor as a marker to determine the new relative anchor based on the indicator value I, which is persistently shared by node a with node b, until the keys used in messages are larger than a *threshold index value* T relative to the new relative anchor. To make the approach highly tolerant to many packets being lost, the value of T is one half the size of an entanglement group, i.e., $T = 2^{R-1}$.

The message header includes a one-bit *key flag* denoted by F to inform the receiver about the transition from the current relative anchor to the next in sequence (i.e., from $A(a)$ to $A(a) + 2^R$). This flag is needed because the indicator I of a jump can only denote large jump values.

Steps at the sender: When node a uses Eq. (3) to jump to a new relative anchor, it stores the value used for I in $IR(a)$, and can start a jump only when its current value of $IR(a)$ is zero, i.e., node a does not carry out multiple consecutive jumps. After node a uses Eq. (4) to obtain the value of k_0 for a message to b, it executes Algorithm 1.

Steps at the receiver: Node b executes Algorithm 2 when it receives a message from node a. $A(b)$ is the current relative anchor assumed by node b; $A(b, J)$ is the stored value of the relative anchor at node b for the last message that b received from a stating $IR(a) = 0$; and f and $IR(b)$ are the stored values at node b of the key flag and the indicator record, respectively.

Algorithm 1 Steps at Sender a

1: $R_K(a) \leftarrow k_0 - A(a)$;
2: **if** $A(a) \leq k_0 < A(a) + T$ **then** $F \leftarrow 1$ **else** $F \leftarrow 0$
3: **if** $k_0 \geq A(a) + T$ **then**
4: $\quad A(a, J) \leftarrow A(a)$; $IR(a) \leftarrow 0$
5: Message $\leftarrow (R(a),\ IR(a),\ F)$

Algorithm 2 Steps at Receiver b

1: **if** $(IR(a) > 0) \wedge (IR(b) \neq IR(a))$ **then**
2: $\quad IR(b) \leftarrow IR(a)$; $A(b) \leftarrow A(b, J) + 2^R(IR(b) \cdot 2^{IR(b)})J$
3: **if** $IR(a) = 0$ **then**
4: $\quad IR(b) \leftarrow IR(a)$;
5: $\quad$ **if** $(F = 1) \wedge (f = 0)$ **then**
6: $\quad\quad f \leftarrow 1$; $A(b) \leftarrow A(b) + 2^R$
7: $k_0 \leftarrow R(a) + A(b)$

Figure 1 illustrates the use of the key flag F. The example assumes that a message has two blocks and $IR(a) = 0$, i.e., the sender did not carry out a jump when computing the value of k_0 for the new message. In the figure, k_1, which is the same as k_0, and k_2 are below the threshold T while k_3 is beyond the threshold T. This does not confuse the receiver, because the value of F applies only to k_0, and the value of k_i selected using Eq. (5). The receiver can safely advance its relative anchor value according to Algorithm 2.

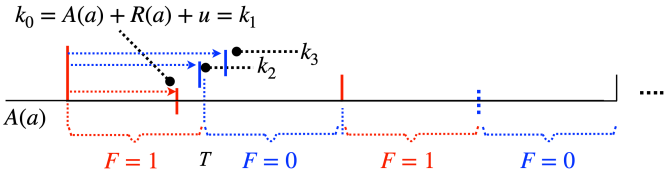


Fig. 1. Use of key flags in the selection of encryption keys

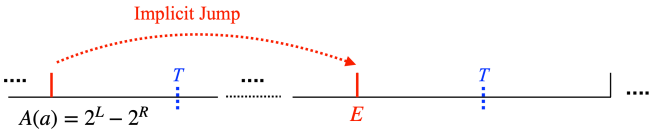


Fig. 2. Reuse of integers in set $\mathcal{I}$ when relative anchor equals $2^L - 2^R$

F. Reusing the Integers from $\mathcal{I}$ in an Entanglement

When the current relative anchor equals $2^L - 2^R$, the next relative anchor to use is E, which remains unchanged. This is illustrated in Figure 2. As the figure illustrates, using E as the new relative anchor amounts to a jump of relative anchors. However, the target value (E) is known and is assumed to be the same as incrementing the value of relative anchor by a

jump of one entanglement group. Accordingly, the CONCISE messages from the sender do not have to indicate the jump explicitly. As the example in Figure 2 illustrates, it is an implicit jump of relative anchor back to E. With this approach, the basis keys used to derive encryption keys always have length L.

G. Using Multiple Entanglements

Each router can maintain multiple entanglements. In the data plane, a router maintains an entanglement with each neighbor router. Furtrhermore, as we have mentioned, CONCISE can be used to secure the signalling of routing protocols [6]. In is the case, the entanglements used to protect the IP packets that flow in the data plane can be different than and independent of the entanglements used to protect routing messages exchanged in the control plane.

H. Point-to-Multipoint Entanglements

Establishing point-to-multipoint cryptographic entanglements in CONCISE is done taking advantage of the ability of receivers to listen to broadcast or multicast addresses and the ability of a sender to jump to new relative anchors without the need for reliable handshakes with intended receivers.

A sender node that needs to send the same messages to all or a subset of its neighbours simply selects a *multipoint relative anchor* for its multipoint entanglement with multiple neighbors. It then transmits its broadcast or multicast message to them to the known multipoint address to which the intended receivers listen. Its message reflects the jump to the multipoint relative anchor, and the rest of the steps are the same as with point-to-point entanglements.

The support point-to-multipoint entanglements applies to messages sent in the control plane as the signalling of routing protocols, as well as IP multicast messages sent in the data plane.

I. CONCISE Message Header

A message exchanged between two trusted nodes using CONCISE includes a CONCISE header used to state the metadata needed by the receiver to decipher the encrypted payload. The elements of the CONCISE header needed to support the functionality we have described are the following:

Relative Key Index or RKI (32 bits): States the value used for $R(a)$.

Indicator of Jump (4 bits): States the value of $IR(a)$.

Entanglement Flags (4 bits): These one-bit flags include the value of the Key Flag (F) set according to Algorithm 1, and the value of the Padding Flag (F_P) set to 1 if padding is used. Two other flags can be used to accommodate various functions used in CONCISE for key selection.

Start of Padding Indicator or SPI (16 bits): States the position of the first bit of padding in the last message block.

Message Timestamp (32 bits): Helps prevent replay attacks by stating the timeliness of the message.

Message Hash (256 bits): Informs the receiver about the integrity of the message. It is a SHA-2 hash that applies to all the header and the original plaintext data.

The placement of the CONCISE header depends on its application, as we describe in the next section.

III. Supporting Secure Packet Forwarding with CONCISE

A. Secure Packet Forwarding within a Computer Network

There are various ways in which CONCISE can be used to support secure packet forwarding within a computer network that uses IP and in which trusted routers are connected to their neighbors with physical links.

One approach consists of using CONCISE as a daemon running at every trusted node that operates between the network layer (i.e., IP) and the link layer. In this approach, the network layer passes an IP datagram to the link layer, which calls the CONCISE daemon to encrypt the entire datagram once the next hop (a trusted node) has been selected at the network layer. The CONCISE header is inserted between the link-layer header and the encrypted IP datagram and the frame is sent to the next hop. At the receiver, the encrypted datagram is passed to the CONCISE daemon for decryption and the Ip datagram is then passed to the IP module. Fig. 3 illustrates this approach. In the figure, the IP header is denoted by H, the link-level header is denoted by LH, and the phyiscal-layer header is denoted by PH.

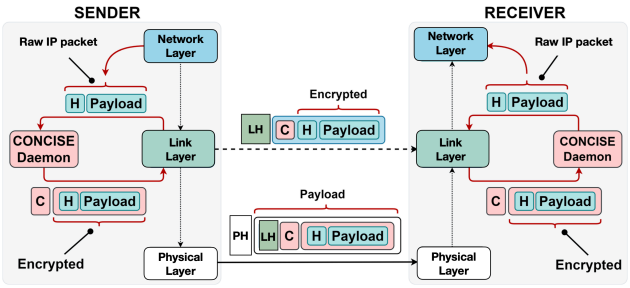


Fig. 3. Using CONCISE to encrypt entire IP packet

Another approach consists of modifying the way in which IPv4 operates in the trusted nodes of a computer network. In this approach, the source (S) and destination (D) addresses and the payload of the IP datagram are encrypted and the rest of the IPv4 header can be sent in the clear. The CONCISE header is included in the IP datagram as an IP option and can be included without encryption. The sender of the IP datagram first determines the next hop for the datagram, and then takes the addresses of S and D and the datagram payload and encrypts them based on the cipher used in the computer network (e.g., an XOR block cipher). The sender uses the first 64 bits of the resulting cipher-text and uses those bits as the IP addresses to include in the IPv4 datagram header. The rest of the cipher-text constitutes the datagram payload. The rest of the IPv4 header is constructed using the payload ciphertext. At the receiver side, the trusted node first deciphers the two cipher addresses and cipher payload using the information stated in the CONCISE. It can then determine the actual source and destination of the datagram, select the next hop and repeat

the encryption step before it forwards the datagram with the encrypted addresses and payload.

B. Secure Packet Forwarding in a VPN

Using CONCISE to establish a VPN managed by the same authority is based on the same encapsulation approach currently used in IPsec in tunnel mode. IP tunnels are established among trusted nodes and encrypted datagrams are sent as payloads of encapsulating IP packets. The encapsulated packets are encrypted and decrypted on a hop-by-hop basis as they traverse the VPN.

At the sender side, the original IP datagram stating the source and destination of the datagram is encrypted in its entirety based on CONCISE. The encapsulating IP datagram states the transmitting and receiving VPN nodes as the source and destination of the datagram. CONCISE is stated as the protocol in the header of the encapsulating IP packet just like IPsec states the Encapsulating Security Payload (ESP) protocol. The CONCISE header follows the IP header just like the ESP header follows the outer IP packet header in IPsec, and the encrypted IP datagram is sent as the payload.

At the receiving end, a VPN node uses the CONCISE header to decrypt the payload. After the payload of the IP packet is decrypted, a VPN router determines the intended destination, selects the next VPN hop, and repeats the steps required to forward the encrypted datagram over the VPN.

C. Comparing CONCISE with IPsec

CONCISE can be used to secure packet forwarding within the same computer network and is simpler alternative to establish a VPN than using IPsec.

Like IPsec, CONCISE provides confidentiality, integrity, and authentication. However, CONCISE is far simpler to configure than IPSec. All trusted nodes are configured with the same two secrets, and the same secrets are used in all the tunnels maintained by a trusted router that is part of the VPN. Unlike IPsec, CONCISE does not require using any other protocol or any handshake to establish, maintain, or disseminate encryption keys once the trusted routers of a VPN are configured. As such, CONCISE is immune to man-in-the middle attacks.

The CONCISE header includes a cryptographic hash of the entire header elements with the bits of all the message blocks before they are encrypted independently of whether the selected cipher performs integrity checks. The hash is also accompanied by a timestamp indicating the time when the hash is computed. Using the hash and its timestamp in CONCISE headers provides a strong tool for integrity checks before the message blocks in the payload are decrypted. If message blocks in the message payload are tampered, a trusted receiver drops the packet.

The hash header element used for integrity check in a CONCISE header can also be used for cryptographic message authentication (independently of the underlying cipher) when combined with the correlated use of entanglement integers that define encryption keys. If an integrity check is passed, it means that the sender is using the same encryption keys as the receiver. The receiver can verify that the sender is who it claims to be based on the correlated use of entanglement integers.

IV. Security Analysis of CONCISE

A. Security Against Brute-Force Attacks

A brute-force attack is one in which adversaries rely on trial and error to find the encryption keys or any useful information that allows them to find the keys eventually. The process can continue until the keys are found. The main defence against these attacks in CONCISE is the secrecy of values and length of integers in set $\mathcal{I}$ from which encryption keys are derived, and for some ciphers the length of encryption keys themselves (L_c). If adversaries manage to somehow discover the key length L_c used in a particular network, they must search through a key space of 2^{L_c} keys. If $L_c \geq 2000$, for example, the search space becomes $2^{1999} \approx 5.75 \times 10^{601}$ keys to find *one key*, which is prohibitive.

B. Security Against Known-Plaintext Attacks

In a known-plaintext attack, adversaries who have access to both plaintext (unencrypted data) and the cipher message (encrypted data) present a target encryption algorithm with the plaintext to study the patterns of encryption emerging in the cipher message with the goal of breaking the encryption algorithm. Adversaries can only gain limited insight on the keys used in an entanglement because the sender selects new keys for each message at random and any given encryption key is not reused for a very long time, which is a function of the key length L_c.

C. Security Against Attacks on CONCISE Operation

Adversaries could attempt to attack CONCISE using the way it operates but without knowing the shared secrets used to configure a network. However, they would face insurmountable difficulty attempting such an attack, and would have no better likelihood of success than with a brute-force attack. This is due to the secrecy of key lengths and the way in which encryption keys are derived from a basis key.

The key or keys used in the cipher to encrypt each message are drawn from integers in set $\mathcal{I}$ and adversaries have no insight as to which integer values of some length L were used as basis keys. This is because basis keys are selected using a function for random increase of integer values in $\mathcal{I}$, such as in the example of Eq. (4), using parameter values that are secret outside an entanglement.

When an XOR block cipher is used, the first challenge for adversaries consists of guessing the length of the keys used for encryption. This results in many thousands of viable choices for the value of L_c used in the block cipher, because the length L of integers in $\mathcal{I}$ are thousands of bits, and the length L_c of encryption keys derived from integers in $\mathcal{I}$ can also be thousands of bits. Therefore, the probability of guessing the correct value of L_c is $P(L_c) \ll 1$.

Given that the encryption keys used to encrypt message blocks are derived using Eq. (5), which is based on the value of the relative anchor and k_0 as the starting point, adversaries must guess the basis key k_0 correctly and then must attempt to obtain the encryption keys used for each message block with no insight on the value of the relative anchor or the jump base. Accordingly, the probability that adversaries decipher the ciphertext resulting from an XOR block cipher is much smaller than $2^{-L_c} \ll 1$ because they must succeed guessing all the encryption keys used in the message, and $L_c \gg 1$.

From the above, it is clear that, independently of the cipher being used in CONCISE, the probability of successfully deciphering a message using the mechanisms defined in CONCISE without knowing the secrets with which a network is configured is insignificant.

V. Performance of XOR Ciphers in an Entanglement

The encryption keys at the sender are available to the cipher with zero delay because the encryption key(s) needed for a message can be usually selected in advance. At the receiver end, the CONCISE header must be processed first to select the encryption keys from the basis key according to the metadata stated in the header. The impact of this step on the performance of the receiver's cipher is minimal because finding encryption keys is of $O(1)$ by the use of the relative key index (RKI) in the header once the CONCISE header is processed.

Once keys are provided to the cipher, the rest of the message encryption and decryption performance is determined by how fast the selected cipher performs. To offer an idea about the performance of XOR ciphers with CONCISE entanglements, we performed an experiment in which two trusted nodes exchange encrypted messages for a duration of 60 seconds with each exchanged message being carried in an IPv4 packet. The payload of each IPv4 packet is composed of a CONCISE header and the bits of the ciphertext with the total payload size of 1500 Bytes to avoid IP packet fragmentation and to exclude the fragmentation overhead from the results.

We ran the sender and receiver nodes in a single computer to remove the impact of network delays for results uniformity. However, one can assume the network delay would be on average in the order of a few milliseconds to a few seconds, depending on the network type and size. We repeated the experiment 10 times for each XOR cipher that we tested, and the results report the average performance of all experiments.

One choice is a simple cipher performing block-based XOR encryption that is most suitable for nodes with resource constraints. This cipher divides the entire message into one or more blocks and each block is of the same size as the encryption keys. The second XOR cipher adds an additional round of XOR to our first cipher for selected blocks. For these two ciphers we used large 2048-bit (256 Bytes) encryption keys.

The average effective processing rate of the first cipher was 2,476.66 Mbps with a standard deviation (SD) of 189.43 Mbps. The effective processing rate of the second cipher was 2,480.53 Mbps with an SD of 100.56 Mbps. These results indicate that CONCISE constitutes a viable approach for supporting secure packet forwarding within a computer network or a VPN managed by the same authority.

VI. Conclusions

We introduced the use of cryptographic entanglements as an alternative to IPsec for the support of secure packet forwarding within a computer network or in a VPN managed by the same authority, and and presented CONCISE as an example of the new approach. A major advantage of CONCISE are that it greatly simplifies the configuration of trusted nodes by allowing all of them to be configured with the same two secrets while establishing and maintaining the correlated use of different encryption keys for different sender-receiver pairs. CONCISE also eliminates the vulnerabilities and complexity resulting from using a key-distribution protocol, relaying on public-key infrastructures, or using cryptographic handshakes subject to man-in-the-middle attacks. Furthermore, CONCISE supports secure point-to-point and point-to-multipoint communication using the same mechanisms, and applies equally well to the data plane as we discuss in this paper and the control plane [6].

CONCISE can use any of the well-known ciphers based on symmetric keys, not just an XOR cipher. The results of simple experiments with XOR ciphers indicate that the speed with which routers would encrypt and decrypt datagrams based on CONCISE is fast enough to support secure packet forwarding in a computer network or VPN managed by the same authority.

Acknowledgments

This work was supported by the Canada Excellence Research Chair in Intelligent Digital Infrastructures at the University of Toronto, funded by the Tri-agency Institutional Programs Secretariat.

References

[1] H. Abbas et al., "Security Assessment and Evaluation of VPNs: A Comprehensive Survey," *ACM Computing Surveys*, Vol. 55, Issue 13s, Dec. 2023.

[2] N. Ferguson and B. Schneier, "A Cryptographic Evaluation of IPsec," Unpublished Manuscript, 1999.

[3] S. Frankel and S. Krishnan, "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap," IETF RFC 6071, Feb. 2011.

[4] J.J. Garcia-Luna-Aceves, "Towards Loop-Free Forwarding of Anonymous Internet Datagrams that Enforce Provenance," *Proc. IEEE Globecom '16*, Dec. 2016.

[5] J.J. Garcia-Luna-Aceves and M. Moghaddassian, "CONCISE: An Efficient and Resilient Alternative to Key Distribution for the Security of Computer Networks," *Proc. IEEE ISNCC'26*, Sept. 2026.

[6] J.J. Garcia-Luna-Aceves and M. Moghaddassian , "Achieving Secure Routing within Autonomous Systems Independently of Routing Protocols," *Proc. IEEE NoF 2026*, Sept. 2026.

[7] C. Kaufman, et al., "Internet Key Exchange Protocol Version 2 (IKEv2)," IETF RFC 7296, Oct. 2014.

[8] S. Kent, "IP Encapsulating Security Payload (ESP)," IETF RFC 4303, Dec. 2005.

[9] S. Savage et al., "Practical Network Support for IP Traceback," *Proc. ACM SIGCOMM '00*, 2000.

[10] A. Snoeren et al., "Hash-based IP Traceback," Proc. ACM SIGCOMM '01, 2001.

[11] F. Tusa, D. Griffin, and M. Rio, "Private Routing in the Internet," *Proc. IEEE HPSR 2021*, June 2021.