Title

CONCISE: An Efficient and Resilient Alternative to Key Distribution for the Security of Computer Networks

Journal

Proc. 13th International Symposium on Networks, Computers and Communications (IEEE ISNCC'26)

Authors

Garcia-Luna-Aceves, J.J.

Moghaddassian, Morteza

Publication Date

2026-09-08

Data Availability

The data associated with this publication are within the manuscript.

Peer reviewed

CONCISE: An Efficient and Resilient Alternative to Key Distribution for the Security of Computer Networks

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*—CONCISE (Cryptosystem for Obfuscation, Non-repudiation and Confidentiality with Integer Set Entanglements) is introduced as an alternative to existing key-distribution methods for the security of computer networks. CONCISE eliminates the need for deploying large numbers of symmetric keys, carrying out cryptographic handshakes, or having to trust third parties that authenticate public keys or distribute keys. Trusted nodes (e.g., routers in an OSPF or EIGRP network) are configured with their own identifiers and the same two secret integers shared by all trusted nodes of a network, which avoids misconfiguration errors associated with complex deployments of secrets. From such limited data any two trusted nodes can become cryptographically entangled in the use of encryption keys. CONCISE can be used with any symmetric-key cipher, including well-known ciphers like the Advanced Encryption Standard (AES) block cipher. It is shown that CONCISE provides very strong security against known attacks, and the performance results of implementations of known ciphers show that CONCISE can be used with any known cipher to provide an efficient approach to secure computer networks.**

I. Introduction

Symmetric-key ciphers like the Advanced Encryption Standard (AES) block cipher are fast and secure by themselves and can be used as part of cryptosystems aimed at securing computer network infrastructures. The cryptosystems currently available rely on different key-distribution approaches, including: configuring each trusted node with all the symmetric keys needed to communicate securely with other nodes, using trusted key-distribution servers to reduce the number of configured symmetric keys, using public-key encryption methods with which trusted nodes can establish session keys with one another (e.g., the Diffie-Hellman key exchange). Unfortunately, as Section II discusses, these approaches suffer from several limitations and vulnerabilities, which range from the misconfiguration or lax configuration of trusted nodes to inherent limitations in the handshakes needed to obtain valid public keys or derive session keys from public keys.

This paper uses the concept of *cryptographic entanglement* as an alternative to known key-distribution methods and as the basis for the design of efficient and resilient cryptosystems for the security of computer networks.

In general, a cryptographic entanglement entails two or more nodes sharing a few secrets different than the encryption keys from which they can reach a state of correlated processing of information. 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.

In this paper we focus on a simple way of implementing cryptographic entanglements, which is to configure in advance all trusted nodes with the same very few secrets through some secure means, and we discuss the benefits of using cryptographic entanglements to secure network infrastructures administered by a single authority.

Section III describes CONCISE (Cryptosystem for Obfuscation, Non-repudiation and Confidentiality with Integer Set Entanglements) as an example of how cryptographic entanglements can be implemented to replace key-distribution methods to secure computer networks.

Trusted nodes in a network using CONCISE are configured with the same two secrets, namely the *Baseline for Entanglement Trust* (BET) and the *jump base*, which are used for the correlated selection and use of encryption keys. 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.

Any cipher based on symmetric keys can be used to take advantage of the cryptographic entanglements established with CONCISE. Depending on the cipher used in CONCISE, one or more encryption keys are chosen by the sender node for each message it sends to the receiver(s) of an entanglement, and different sequences of encryption keys are used for different entanglements.

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 IV discusses the small overhead incurred by the maintenance of cryptographic entanglements in CONCISE and presents experimental results showing the performance of well-known 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 by all ciphers are above 2.4 Gbps, which indicates that CONCISE can be used to secure the signaling of any communication protocol, and routing protocols in particular.

Section V shows that CONCISE is secure against known attacks, and Section VI discusses CONCISE security features, namely: simple configuration of trusted nodes, which prevents misconfiguration errors, confidentiality while eliminating the vulnerabilities of current key-distribution approaches, and integrity and authentication enforced by the headers of ciphertexts. Section VII outlines examples of how CONCISE can be used in networks managed by the same authority.

II. Limitations of Key Distribution Methods

Securing the communication protocols of a computer network requires the use of symmetric or asymmetric cryptography to support authentication and confidentiality (e.g., [1], [12], [16], [19], [20]). In the context of a network infrastructure, even in the case of an overlay network, using symmetric keys in trusted nodes is needed because the encryption and decryption of messages are much faster than using public-key cryptography.

In theory, symmetric-key ciphers like AES and the XOR block cipher are fast and secure given a proper deployment of encryption keys. However, their usefulness is hindered in practice by the limitations of existing key-distribution methods used to provide trusted nodes with symmetric keys. The following paragraphs discuss the limitations of current approaches.

A. Deploying and Managing Encryption Keys

The deployment of symmetric keys to trusted nodes in a network has been based on two main approaches, namely: assigning symmetric keys manually and assigning public-private key pairs to individual nodes to protect the exchange of symmetric encryption keys by infrastructure nodes. Unfortunately, there are major challenges in both approaches.

Manually assigning symmetric keys to pairs of trusted nodes and configuring them is very challenging and prone to attacks resulting from lax configurations or configuration errors. Clearly, if all network nodes need to communicate with each other, every trusted node must hold at least one symmetric key with every other trusted node in the network. This does not scale as the number of trusted nodes n increases, because $n \times (n-1)/2$ symmetric keys must be used in the worst case. The configuration problem remains challenging even when trusted nodes need to communicate with only a small subset of other trusted nodes. With n nodes that communicate with at most m neighbor nodes, a network administrator must handle up to $n \times m/2$ symmetric keys, because the keys used in the network should be different for different pairs of nodes to avoid known attacks (e.g., [2], [17]).

To make the problem worse, any changes in the network infrastructure (e.g., removing or adding nodes) or refreshing keys require many changes to the symmetric keys that are currently used in the infrastructure, which presents major scalability challenges, as has been noted in [16]. If changes are not reflected on a timely manner, keys can be wrongly associated with nodes that no longer use them, which enables adversaries to leverage them for planning strategic attacks (e.g., impersonation attacks [9]).

In theory, symmetric keys can be assigned using automated key distribution methods [10], [14]. However, the transfer of symmetric keys to trusted nodes must be protected, and hence automated key distribution systems typically require devices to use public-private key pairs [14] that obviates the use of pairwise assignment of symmetric keys in the first place.

Using public keys presents its own deployment and key management problems. Trusting public keys requires digital certificates that sign them, and certificate authorities that verify the signed certificates. This requires every trusted node to obtain a digital certificate for its own public key [19], which is costly and ineffective in a large network. Equally problematic is the fact that digital certificates require implicit trust in certificate authorities, which is prone to known potential security risks [19]. If public-private key pairs are used, infrastructure devices communicating with each other must hold a copy of each other's public key [16]. Even if a trusted node needs to communicate with only a subset of the other network nodes, it must be configured with or find the public keys of the nodes with which it communicates, which requires either careful configuration or secure end-to-end communication with a third trusted party.

B. Deriving and Relying on Static Symmetric Keys

Symmetric keys for fast encryption and decryption can be derived from public keys that are, in theory, easier to deploy and maintain. Different approaches have been proposed and used for this purpose. The best-known example of this is the Diffie-Hellman (DH) key exchange protocol [15], and several others have been proposed, such as the Elliptic Curve Diffie–Hellman (ECDH) protocol [15]. Despite the appealing nature of key-exchange protocols, they have a number of limitations, including: The lack of forward secrecy in long-term use of keys, susceptibility to man-in-the-middle attacks and replay attacks, and requiring heavy computations to calculate the session keys, which may not be suitable for deployments in which trusted nodes have limited computing power [3].

In addition, symmetric keys that are derived through key exchange handshakes typically remain static for a long time

before they are refreshed or expire, which leaves all ciphertexts produced by the same key exposed to adversarial attacks leading to the discovery of the key. This problem is usually addressed in two ways. First, by generating sequential keys that are all produced from the same master key, and second, by key refreshment methods that update the encryption keys on a timely basis, such as the methods used in TLS. Unfortunately, the key refreshment process is not very secure in either scenario.

III. CONCISE

CONCISE replaces the traditional approaches for key distribution and takes advantage of any symmetric-key cipher. CONCISE operates by establishing and maintaining cryptographic entanglements among nodes using common functions and two secrets shared among all trusted nodes of a computer network. Although trusted nodes share the same two secrets, each cryptographic entanglement is defined between a sender and one or multiple receivers. To ensure that the state of correlated use of encryption keys is maintained, a ciphertext sent to receivers consist of a header and a payload, and the header provides hints about the encryption keys used by the sender so that receivers can decipher the payload even when messages are lost or delivered out of order.

CONCISE can be used to secure the message payloads sent by any communication protocol in the protocol stack. This can be done as part of a communication protocol itself or as an external daemon called by the communication protocol.

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

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

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

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

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:

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

The variable I in Eq. (2) 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.

Figure 1 illustrates the use of a jump to change the relative anchor based on Eq. (2). In the example, node a executes a jump of $(I \cdot 2^I)J$ entanglement groups with its first message to node b. The example also illustrates the fact that the hint provided by node a to node b regarding the value of the first key used for the encrypted message in the payload is based on the agreement that a and b maintain on the relative anchor used as a reference point for the values of $R(a)$ and $IR(a)$ stated in the message header. The meaning of variable F in the figure is described below.

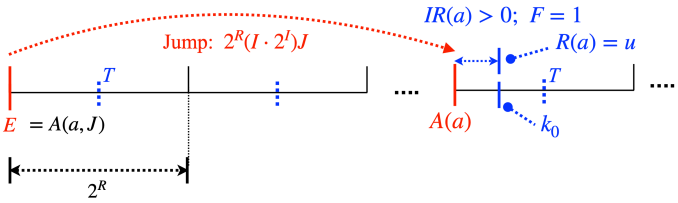

Fig. 1. A jump to a new relative anchor

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. (3).

$$k_0 = \begin{cases} A(a) + R_K(a) + u & \text{if } \ IR(a) = 0 \\ A(a) + u & \text{if } \ IR(a) > 0 \end{cases} \qquad (3)$$

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

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:

$$k_1 = \mathcal{L}_{L_c}[k_0] \qquad (4)$$

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

If the cipher used in a deployment of CONCISE requires a single encryption key per message, which is the case of AES for example, then k_1 is simply the number of least significant bits of k_0 that match the key length used in the cipher. If the length of integers in set $\mathcal{I}$ is the same as the key length needed in the cipher used in CONCISE, then $k_1 = k_0$.

E. Correlating Keys and Jumps without Reliable Handshakes

CONCISE does not do any retransmissions and does not use any handshakes for the sender and receivers of an entanglement to agree on encryption keys. 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.

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: $k_0 \leftarrow R(a) + A(b)$

Steps at the sender: When node a uses Eq. (2) 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. (3) 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.

Figure 2 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. (4). The receiver can safely advance its relative anchor value according to Algorithm 2.

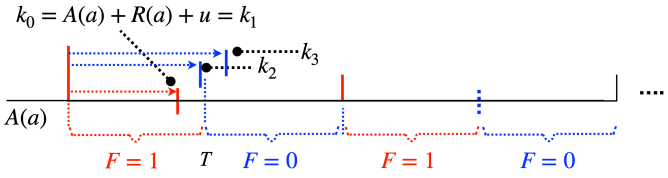

Fig. 2. Use of key flags in the selection of encryption keys

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 3.

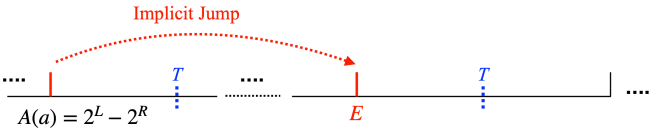

Fig. 3. Reuse of integers in set $\mathcal{I}$ when relative anchor equals $2^L - 2^R$

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 3 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. Point-to-Multipoint Entanglements

Establishing point-to-multipoint cryptographic entanglements in CONCISE is remarkably simple by 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.

IV. Overhead of Cryptographic Entanglements

The overhead incurred by the adoption of cryptographic entanglements consist of the transmission overhead incurred by the headers used as part of ciphertexts, and the time required to encrypt and decrypt messages based on the symmetric-key cipher chosen for a given network.

A. 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 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 CONCISE header uses the first 43 bytes of the message payload with the rest of the payload left for communicating encrypted data. Some ciphers may also apply padding on the payload.

B. Cipher Performance in an Entanglement

Trusted nodes in CONCISE have the freedom of using any symmetric-key cipher for using encryption keys provided through the 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 as once the CONCISE header is processed, finding encryption keys is of $O(1)$ by the use of the relative key index (RKI) in the header.

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 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. For each cipher that we tested, we repeated the experiment 10 times, and the results report the average performance of all experiments.

We chose four different ciphers to cover a wide range of applications. Our first 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. Our second 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. For the two other ciphers in our experiment, we used the AES-CBC and AES-GCM variations of the Advanced Encryption Standard (AES) with both using 256-bit (32 Bytes) encryption keys [18]. These ciphers are more commonly used with hardware built for standard nodes.

Fig. 4 shows the average effective processing rate of each cipher, which is the rate at which the receiver can decrypt the encrypted IP packet payload. The results show that the rates are substantially high and above 2.4 Gbps across all ciphers in the experiment.

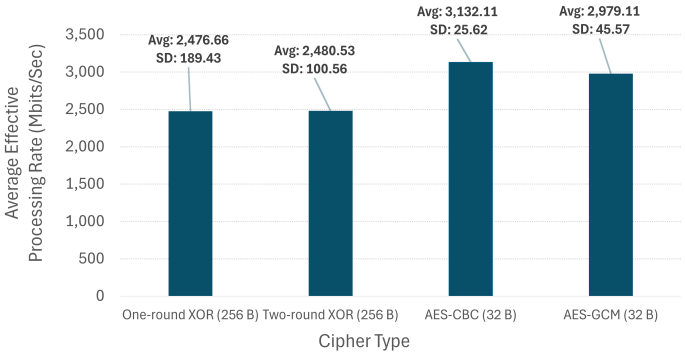

Fig. 4. Performance comparison of four ciphers using entangled keys

We believe that there are two main factors that make AES faster than the XOR-based ciphers in this experiment. One factor is the memory-bound operation needed to split a message into multiple blocks in the XOR-based cipher. The other factor is that multiple encryption keys are used for encrypting each IPv4 packet with XOR-based ciphers, while AES encryption consumes only one encryption key for the entire packet payload.

The results also indicate that AES-CBC was slightly faster than AES-GCM in our experiment. This is mainly because AES-CBC does not provide message authentication and AES-GCM does, which removes the corresponding processing overhead in AES-CBC. If message authentication were added to AES-CBC, the AES-GCM would show better performance.

We also measured the average CPU and Resident Set Size (RSS) for each of the sender and receiver processes in our experiment where our results indicate all ciphers require minimal CPU and memory for encryption and decryption of the IPv4 packets with AES-based ciphers to require more memory. RSS shows the portion of the memory that is occupied by a process in the physical memory (RAM). We note that in our experiments the memory required to store entangled encryption keys is excluded form the results for uniformity, because it depends on keys sizes. However, we note that all keys are derived from only two integers ($\mathcal{B}$ and J) that all trusted nodes share in the network.

Cipher	Average CPU Utilization (%)	Average RSS (KB)
One-Round XOR (256B)	R: (Avg:1.57, SD:0.23) S: (Avg:4.16, SD:0.02)	R: (Avg:3,573.3, SD:24.79) S: (Avg:3,563.2, SD:26.28)
Two-Round XOR (256B)	R: (Avg:1.57, SD:0.09) S: (Avg:4.16, SD:0.01)	R: (Avg:3,569.6, SD:28.13) S: (Avg:3,574.8, SD:17.53)
AES-CBC (32B)	R:(Avg:0.92, SD:0.07) S: (Avg:4.17, SD:0.02)	R: (Avg:7,331.6, SD:19.14) S: (Avg:7,273.2, SD:17.62)
AES-GCM (32B)	R: (Avg:1.09, SD:0.03) S: (Avg:4.19, SD:0.02)	R: (Avg:7,401.2, SD:22.13) S: (Avg:7,530.4, SD:20.49)

TABLE I
AVERAGE CPU AND RSS UTILIZATION FOR CIPHERS USING ENTANGLED KEYS (R IS USED FOR RECEIVER, S FOR SENDER, AVG FOR AVERAGE, AND SD FOR STANDARD DEVIATION)

V. 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 defense 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. Even for relatively small key lengths (e.g., 128, 192 or 256 bits as in AES), the search space is very large.

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. (3), using parameter values that are secret outside an entanglement.

If AES is used as the cipher in CONCISE, a symmetric key corresponding to the 128, 192, or 256 least-significant bits of a basis key k_0 is used for each message being sent in an entanglement. Given that k_0 is chosen randomly by the sender, the probability of adversaries deciphering a ciphertext is the same as in AES using a random symmetric key.

For the case in which 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. (4), 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.

VI. Security Features of CONCISE

A computer network with trusted nodes using CONCISE enjoys ease of configuration, strong confidentiality, integrity, and cryptographic authentication.

Ease of Configuration: This is a major motivation for the design of CONCISE, but is not a security feature. However, the simplicity of configuring trusted nodes helps prevent attacks due to misconfiguration or lax configuration. All trusted nodes are configured with the same two secrets, rather than requiring different secrets for different network links as it is the case in OSPFv2, for example.

Confidentiality: CONCISE allows trusted nodes to exchange messages with payloads that are always encrypted, without the need for any handshake to establish trust or disseminate keys. As such, it is immune to man-in-the middle attacks. Current alternatives require trusted nodes (e.g., routers) to be configured with static symmetric keys per neighbor, or the use of mathematically complex ways of generating static symmetric keys from public-private key pairs. As we have shown, CONCISE provides strong confidentiality, and Section IV shows that the cryptographic approach used in CONCISE is very fast.

Integrity: The CONCISE header accompanies 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.

Message Authentication: 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.

VII. Using CONCISE in Networks

The advantages that CONCISE offers over key distribution derive from the ease of configuration enabled by cryptographic entanglements and the elimination of the vulnerabilities associated with current key-distribution methods. We discuss three examples of using CONCISE in networks managed by a single administrative authority to highlight these advantages.

Routing within Autonomous Systems: The routing protocols used today within the Autonomous Systems (ASes) of the Internet send signaling messages in the clear. They do not support confidentiality, which makes them vulnerable to many passive and active attacks without the need to compromise the routers themselves. Routing protocols like OSPF, RIP, and EIGRP can provide message integrity and authenticity by relying largely on manual configurations of many symmetric keys and passwords, or the use digital certificates and automated key-distribution methods that are known to be vulnerable to attacks associated with Public-key Infrastructures (PKIs) and digital certificates.

CONCISE can be used to secure any routing protocol running within an autonomous system (AS) without requiring any changes to the routing protocols themselves. This is done by establishing cryptographic entanglements among trusted routers using the common baseline for entanglement trust and

jump base with which they are configured. Entangled routers exchange only encrypted routing messages over any insecure link or network, and the transmission overhead per routing message is negligible (43 bytes). This prevents attackers from accessing any routing information from the routing messages being exchanged; furthermore, unlike IPsec, CONCISE does not require using digital certificates and a PKI, and is consequently not prone to their associated security threats.

The details of how CONCISE can be used in this way are presented in [7].

Tactical ad-hoc networks: Securing this type of networks is exceptionally difficult [4], [21] because they lack a centralized infrastructure, which requires the nodes themselves to enforce security without being able to rely on third parties and certificate authorities. All prior approaches to secure ad-hoc networks rely on the use of symmetric keys or public-private key pairs. Distributing and managing symmetric keys for all pairs of communicating nodes or managing public keys distributively become untenable as the network size increases.

CONCISE can provide an alternative that is simple to deploy and can be used to protect existing routing protocols for ad-hoc networks, as well as the network control and data planes by encrypting every massage payload. CONCISE is particularly attractive for tactical applications in which nodes can be configured prior to deployment.

Overlay Networks: CONCISE is very attractive for overlay networks because it can secure the messages exchanged between overlay routers in the data and control planes, without requiring complex configuration of the tunnels needed for the overlay or changes to the routing protocols used over it. Similar benefits apply to its application to virtual private networks [1].

VIII. Conclusions

We introduced the use of cryptographic entanglements as an alternative to existing key-distribution methods, and presented CONCISE as an example of the new approach. The key advantages of replacing current approaches for key distribution with CONCISE is 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. Furthermore, CONCISE eliminates the vulnerabilities and complexity resulting from having to trust key-distribution servers, relaying on public-key infrastructures, or using cryptographic handshakes subject to man-in-the-middle attacks.

We discussed the main advantages of using CONCISE to secure networks administered by the same authority in which trusted nodes are configured with the same two secret integers to establish a state of correlated selection and use of encryption keys. CONCISE can use any of the well-known ciphers based on symmetric keys, and the results of our experiment comparing the performance of a few ciphers using keys from an entanglement demonstrate that CONCISE can easily support per-hop encryption and decryption and be used to secure any communication protocol, and routing protocols in particular.

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] L. Åkerberg and V. Johansson Baurne, "Practical Analysis and Simulation of OSPF Protocol Vulnerabilities," Tech. Rep., KTH Royal Institute of Technology, Sept. 2024.

[3] F. A. Alaba et al., "Internet of Things Security: A survey," *Journal of Network and Computer Applications*, Vol. 88, pp. 10–28, 2017.

[4] A. Boukerche (Ed.), *Handbook of Algorithms for Wireless Networking and Mobile Computing*, Chapman & Hall/CRC, 2006.

[5] I. Butun, P. Österberg, and H. Song, "Security of the Internet of Things: Vulnerabilities, Attacks, and Countermeasures," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 616–644, 2019.

[6] E. Demirci and E. Olca, "A Review of Grover's Search Algorithm: Applications and Modifications," *Proc. 9th International Artificial Intelligence and Data Processing Symposium (IDAP)*, 2025.

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

[8] D. Dinev and G. Spasova, "Benchmarking and Teaching Symmetric Cryptography: A Comparative Study of Algorithms and Cryptanalysis Methods with an Educational Simulator," *Proc. CIEES 2025*, 2025.

[9] V. Hassija et al., "A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures," *IEEE Access*, vol. 7, pp. 82 721–82 743, 2019.

[10] A.M. Hegland et al., "A Survey of Key Management in Ad Hoc Networks," *IEEE Communications Surveys & Tutorials*, Vol. 8, No. 3, 2006.

[11] V.-H. Hoang, E. Lehtihet, and Y. Ghamri-Doudane, "Password-Based Authenticated Key Exchange based on Signcryption for the Internet of Things," *Proc. IEEE 2019 Wireless Days (WD)*, 2019.

[12] G. Huston, M. Rossi, and G. Armitage, "Securing BGP — A Literature Survey," *IEEE Communications Surveys & Tutorials*, Vol, 13, No. 2, 2011.

[13] D. E. Kouicem, A. Bouabdallah, and H. Lakhlef, "Internet of Things Security: A Top-Down Survey," *Computer Networks*, vol. 141, pp. 199–221, 2018.

[14] I. Mansour et al., "Secure Multihop Key Establishment Protocols for Wireless Sensor Networks," *Proc. International Conference on Cryptography and Security Systems*, 2014.

[15] M. R. Mishra and J. Kar, "A study on Diffie-Hellman Key Exchange Protocols," *International Journal of Pure and Applied Mathematics*, Vol. 114, no. 2, 2017.

[16] S. K. Mousavi, A. Ghaffari, S. Besharat, and H. Afshari, "Security of Internet of Things based on cryptographic algorithms: a survey," *Wireless Networks*, vol. 27, no. 2, pp. 1515–1555, 2021.

[17] G. Nakibly et al., "Persistent OSPF Attacks," *Proc. NDSS Symposium 2012*, Feb. 2012.

[18] D. Sarangi, "A comparative Study of AES Encryption Modes and Hashing for Blockchain Applications," National College of Ireland, 2024.

[19] K.-A. Shim, "A Survey of Public-Key Cryptographic Primitives in Wireless Sensor Networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 577–601, 2015.

[20] B. Vetter, F. Wang, and S.F. Wu, "An Experimental Study of Insider Attacks for OSPF Routing Protocol," *Proc. IEEE ICNP '97*, Oct. 1997.

[21] H. Yang et al., "Security in Mobile Ad Hoc Networks: Challenges and Solutions," *IEEE Wireless Communications*, Feb. 2004.