Chapter 7

MULTI-CHANNEL SECURITY THROUGH DATA FRAGMENTATION

Micah Hayden, Scott Graham, Addison Betances and Robert Mills

Abstract This chapter presents a novel security framework developed for a multi-channel communications architecture that achieves security by distributing messages and their authentication codes across multiple channels at the bit level. This method of transmission provides protection from confidentiality and integrity attacks without relying on encryption. The two communicating parties utilize existing key exchange mechanisms to pass initialization information. The framework operates by assigning to each message bit a fragment identifier using a hardware-based stream cipher as a pseudorandom number generator, and transmitting specific message fragments across each channel. This prevents the entirety of a message from being transmitted over a single channel and spreads the authentication across the available channels, enabling the sender and receiver to identify a compromised channel even in the presence of a sophisticated man-in-the-middle attack where the adversary forces message acceptance at the destination, perhaps by altering the message error detecting code. Under some conditions, the receiver can recover the original message without retransmission. The holistic framework is attractive for critical infrastructure communications because it provides availability while defending against confidentiality and integrity attacks.

Keywords: Multi-channel communications, security, data fragmentation

1. Introduction

Traditional communications frameworks rely on information traveling over a single communications link. Methods exist for communicating administrative information separately from message data; however, if an adversary gains access to the communications link carrying the message data, he/she can obtain the entire message content. This forces the use of encryption to protect the confidentiality of the transmissions. Typi-

J. Staggs and S. Shenoi (Eds.): Critical Infrastructure Protection XIV, IFIP AICT 596, pp. 137–155, 2020.
https://doi.org/10.1007/978-3-030-62840-6_7

cally, these systems use a hashed message authentication code (MAC) to check the integrity of each message and retransmit a message if necessary. However, such methods are susceptible to adversarial action that fools a receiver to accept an invalid message. The guarantees of confidentiality and integrity are of paramount importance in critical infrastructure communications due to the operational and physical impacts of successful attacks.

This chapter extends the research of Wolfe et al. [12], which proposed the use of two channels to defeat various adversarial actions. The effort, which targeted low-power devices, addressed each type of attack with an individual security policy. In contrast, this chapter proposes a tunable framework for multi-channel communications, enabling a user to address multiple types of attacks simultaneously. The architecture utilizes data fragmentation and duplication to provide increased security and reliability. By splitting the data into fragments at the bit level and distributing the fragments over a channel set, information leakage is reduced in the presence of adversarial actions. Additionally, man-in-the-middle attacks can be detected and defeated even if an adversary is able to modify the error correcting code to fool the receiver. As expected, there is an overhead associated with these services. However, due to the tunable nature of the architecture, a trade-off can be struck between the services provided and the overhead involved.

The proposed secure communications framework is intended to serve as a road map for network designers to create multi-channel communications systems for specific use cases. Indeed, security protocols such as Transport Layer Security (TLS) could reasonably incorporate multi-channel communications to provide security when multiple lines of communication are available.

2. Background

This section describes security developments that are relevant to multi-channel communications systems. As customary, Alice and Bob are the communicating entities and Eve is the adversary.

Wolfe et al. [12] have proposed multi-channel communications as a viable security alternative for low-power devices. They describe how multiple channels can thwart eavesdropping attacks by splitting the data across the channels and defeat integrity attacks by duplicating the data across two or more channels. However, they do not mention specific mechanisms for splitting and duplicating the data, nor do they perform the two tasks simultaneously. This research extends the work of Wolfe and colleagues by proposing a mechanism for splitting messages across multiple channels. Also, it identifies several tunable characteristics that can meet the security requirements.

2.1 CIA Triad

The confidentiality, integrity and availability (CIA) triad covers the key security requirements. User-specific confidentiality, integrity and availability needs dictate the level and type of security required for a given application:

- **Confidentiality:** The confidentiality requirement specifies that only the sender and intended recipient(s) may correctly decode/decrypt the transmitted data.

- **Integrity:** The integrity requirement specifies that the received data is correct and unmodified.

- **Availability:** The availability requirement specifies that the data arrives within a certain time or latency window. This is typically quantified using traditional quality of service (QoS) metrics.

2.2 Transport Layer Security

Transport Layer Security 1.0, which was specified in RFC 2246, provides communications privacy over the Internet – it "[allows] client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering or message forgery" [4]. It is a widely-used protocol for protecting communications. RFC 2246 specifies the following four goals for the Transport Layer Security Protocol in order of priority:

- **Cryptographic Security:** The protocol should be used to establish a secure connection between two parties.

- **Interoperability:** Independent programmers should be able to develop applications using the protocol that will successfully exchange cryptographic parameters without any knowledge of each other's code.

- **Extensibility:** The protocol should provide a framework into which new public key and bulk encryption methods can be incorporated as necessary.

- **Relative Efficiency:** Since cryptographic – especially key – operations tend to be very computationally intensive, the protocol must incorporate an optional session caching scheme to reduce the number of connections that need to be established from scratch.

The Transport Layer Security Protocol achieves these four goals by relying on the TLS Record Protocol for connection security and the TLS Handshake Protocol to authenticate the two parties.

The TLS Record Protocol client specifications list two basic properties [4]:

- **Private Connections:** Symmetric cryptography is used for data encryption. A unique symmetric key is generated for each connection and key transfer is accomplished using a secret negotiated by another protocol (e.g., TLS Handshake Protocol).

- **Reliable Connections:** Message transport includes a message integrity check using a keyed message authentication code.

The TLS Record Protocol allows the encapsulation of various higher-level protocols. One of these protocols is the TLS Handshake Protocol, which guarantees authentication, confidentiality, integrity and availability.

Deprecation of Secure Sockets Layer. Transport Layer Security became the *de facto* protocol for securing transport layer communications after the deprecation of Secure Sockets Layer Version 3 (SSLv3) described in RFC 7568 [1]. The SSLv3 key exchange mechanism and cipher suites were attacked over several years, leading to the creation of Transport Layer Security 1.0 and 1.1 specified in RFC 2246 [4] and RFC 4346 [5], respectively. However, there was no widespread support of these replacement protocols, which led to the continued use of SSLv3 [1].

Starting with Transport Layer Security 1.2 in RFC 5246 [6], backwards compatibility with SSL was eliminated to ensure that sessions would not support the negotiation and use of SSL security. In fact, RFC 7568 [1] states that "SSLv3 is comprehensively broken." Specifically, it has flaws in its cipher block chaining (CBC) modes and weaknesses in its stream ciphers. Key exchange is vulnerable to man-in-the-middle attacks through two methods – renegotiation and session resumption. Moreover, it relies on SHA-1 and MD5 hashing, which are considered weak and are being replaced with stronger hash functions. Transport Layer Security 1.2 addresses all these weaknesses using new cryptographic methods and features. RFC 7568 states that SSLv3 must not be used, indicating a complete shift to Transport Layer Security 1.2.

Transport Layer Security Development. Transport Layer Security 1.0, which was defined in 1999 [4], did not indicate significant shifts from SSL. Specifically, it allowed for the negotiation of SSL connections.

The first major shift was made in Transport Layer Security 1.1, which addressed several SSL vulnerabilities. The changes were [5]:

- Replacement of the implicit initialization vector with an explicit initialization vector for protection against cipher block chain attacks.
- Handling padding errors to protect against cipher block chain attacks.

The principal goals and properties of the Transport Layer Security Protocol remained the same from Transport Layer Security 1.0 through 1.2. However, Transport Layer Security 1.2 incorporates several changes from Transport Layer Security 1.1. It allows for improved flexibility, specifically in negotiating cryptographic algorithms and specifying cipher-suite-specific pseudorandom functions. There is support for authenticated encryption and additional data modes. Transport Layer Security 1.2 eliminates support for cipher suites such as IDEA and DES. Finally, it lowers the support for SSLv2 backwards-compatibility from a "should" to a "may," under the assumption that it will become a "should not" in the future [6].

As the security environment continued to develop, written standards were required to ensure that entities communicate security parameters via the same language to ensure clarity and efficient communication. These guidelines were specified in RFC 3552: "The Guidelines for Writing RFC Text on Security Considerations" [11]. Transport Layer Security 1.3 incorporates the security updates from Transport Layer Security 1.2 and a change in the protocol goals to align with the language specifications in RFC 3552. The updated goals are [10]:

- **Authentication:** The server side of a channel is always authenticated whereas the client side is optionally authenticated. Authentication can occur via asymmetric cryptography or a symmetric pre-shared key.
- **Confidentiality:** Data sent over a channel after establishment is only visible to the endpoints. Transport Layer Security does not natively hide the length of the data it transmits, although endpoints are able to pad Transport Layer Security records in order to obscure lengths and enhance protection against traffic analysis.
- **Integrity:** Data sent over a channel after establishment cannot be modified by attackers without detection.

The major changes incorporated in Transport Layer Security 1.3 reflect significant research in secure communications [10]. The protocol modifies the cipher suite concept to separate the authentication and key exchange mechanisms from the record protection algorithm. It prunes the list of allowable cipher suites by removing legacy algorithms and eliminating the static RSA and Diffie-Hellman cipher suites, instead allowing only public-key mechanisms that provide forward secrecy – the assurance of the secrecy of past sessions even if future sessions are compromised. It requires handshake messages to be encrypted and restructures the handshake state machine to be more consistent and remove overhead.

Transport Layer Security 1.0 has been adapted to the current Transport Layer Security 1.3 to keep up with new vulnerabilities and attacker capabilities. The pattern demonstrates the willingness to adapt to an ever-changing security environment by developing new methods and protocols that maintain secure communications. The adaptation is expected to continue, including providing support for multi-channel communications in the coming years.

2.3 Data Fragmentation

An efficient and cryptographically secure method should be used to fragment messages. A linear feedback shift register (LFSR) provides an elegant way of realizing long, pseudorandom sequences with minimal software/hardware requirements, making it an ideal candidate for data fragmentation. An LFSR has a series of flip flops and a feedback path that outputs a single bit of output during each clock cycle.

The maximum output length of an m-bit LFSR is given by [9]:

$$Length = 2^m - 1 \tag{1}$$

After $2^m - 1$ values, the sequence repeats itself; this length is the period of the LFSR.

Paar and Pelzl [9] provide a proof that an LFSR can be broken with $2m$ key stream bits due to the linear progression of its internal state. This leads to the Trivium hardware-oriented synchronous stream cipher [3]. By chaining three LFSRs, the internal state of each LFSR does not evolve in a linear fashion.

De Canniere and Preneel [3] describe the construction of the Trivium cipher and its hardware requirements at the gate level, and provide a brief security analysis. The stream cipher has an output period of 2^{64} bits. It also has low-power hardware implementations. For these reasons, the proposed framework leverages the Trivium cipher to generate pseudorandom sequences to map each message bit to a corresponding fragment identifier.

2.4 Diffie-Hellman Key Exchange

A communications system must guarantee security as long as its initialization/key information are kept secret. Thus, a secure method is needed to exchange the system initialization parameters over an insecure channel.

The Diffie-Hellman Key Exchange algorithm is a one-way function that relies on the commutative property of exponentiation. The algorithm, which incorporates setup and key exchange phases, guarantees that only Alice and Bob can obtain the session key from the transmitted information, even if Eve is able to access all communications. Details about the algorithm and a proof of its security are provided in [9]. The Diffie-Hellman Key Exchange algorithm is used to exchange initialization information in the proposed data fragmentation scheme. Alice and Bob both compute the session key k_{AB} and utilize the most significant bits to generate the key and initialization vector for the Trivium cipher.

2.5 Regulatory Standards

The North American Electric Reliability Council (NERC) created the Critical Infrastructure Protection (CIP) Security Standards CIP-002-014 [8] to formalize security requirements for the entire energy sector, ranging from personnel and training requirements in CIP-004-6 to information protection outlined in CIP-011-2, which is the standard adopted in this work. CIP-011-2 seeks to prevent unauthorized access to information about the bulk electric system and specifies requirements for protecting cyber systems against compromises that could lead to misoperation or instability.

Fries and Falk [7] reference the International Electrotechnical Commission IEC 62443-3-3 Standard that imposes two requirements directly related to secure communications:

- **Requirement 3.3.1 Communications Integrity:** The control system shall provide the capability to protect the integrity of transmitted information.

- **Requirement 4.4.1 Communications Confidentiality:** The control system shall provide the capability to protect the confidentiality of information at rest and in remote access sessions traversing an untrusted network.

2.6 Summary

There is clearly a vested interest in developing mechanisms that ensure the confidentiality and integrity of communications, specifically in the critical infrastructure. Certain regulatory standards specify the requirements for secure communications. The Transport Layer Security protocol is one of the primary methods for securing networked communications. The protocol has gone through several iterations to accommodate the changing security needs as reflected in the three main goals of Transport Layer Security 1.3, namely authentication, confidentiality and integrity. Other ongoing work addresses the emerging field of multi-channel communications. This research proposes a data fragmentation scheme based on the Trivium cipher that distributes message content across multiple channels.

3. Proposed Framework

This section discusses the proposed framework, including its goals, tunability and operation,

3.1 Goals

The proposed framework is designed to accomplish two goals: (i) explore the challenges in a multi-channel security system; and (ii) investigate the potential security services obtained through its use. A full communications session from initialization through message receipt is completed across a user-specified number of channels. Adversarial actions can occur on any of the available channels to demonstrate resilience. This gives an increased understanding of the effort required to field an operational system. Abstractions are used to reflect issues that are yet to be resolved, but developmental paths or guides are provided for future research. The information an adversary gains from a given attack against a single channel in an unencrypted scenario is specified; in an encrypted scenario, the proposed multi-channel framework would likewise use encryption. The comparison demonstrates how a multi-channel architecture can identify, mitigate and even defeat several adversarial attacks.

3.2 Tunability

A user of the framework would determine several parameters based on a set of security requirements. The parameters include the number of channels, duplication factor, number of fragments and a fragment-to-channel mapping. Each of these parameters identifies a trade-off between the elements of the confidentiality, integrity and availability triad and the associated overhead.

A channel requires a handshake to initialize its connection and an associated network interface. As the number of channels increases, more

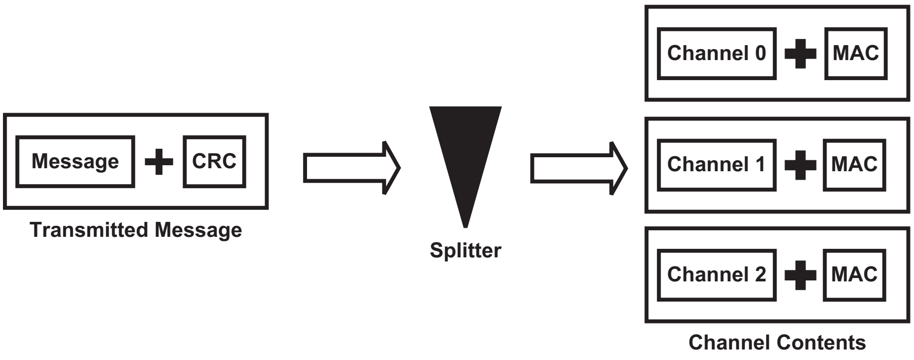

Figure 1. Three-channel message transmission.

channels are available for an adversary to target, but the defender also gains the ability to increase obfuscation.

The duplication factor strictly refers to the duplication of information. No duplication indicates that only a single copy of each bit of message is sent; this provides the maximum level of confidentiality because all messages on all channels must be intercepted in order to fully recreate the message. Full duplication indicates that the entire message is sent across each of the available channels, providing the maximum level of availability but with reduced confidentiality. As the duplication factor increases, there is a corresponding increase in the amount of information sent.

A single fragment could represent a single bit of a message while the maximally-sized fragment could contain the entire message. As the number of fragments increases, greater data obfuscation is provided by spreading the fragments across the channel set. However, there is additional overhead because more key bits are required to map a message bit to a fragment. This illustrates the complex environment of a multi-channel communications architecture as well as the unique advantage it presents to users. A user of the framework may specify the desired services but, and in doing so, would accept the incurred overhead.

Figure 1 shows the basic mechanism for a three-channel transmission. To transmit a message, the system computes a cyclic redundancy code (CRC), appends it to the end of the message and then assigns each bit to a channel. Each channel transmits its own set of data with a corresponding channel CRC.

Figure 2 shows the receiving side of the communications session. Each channel CRC is checked to verify the channel contents. Next, each bit

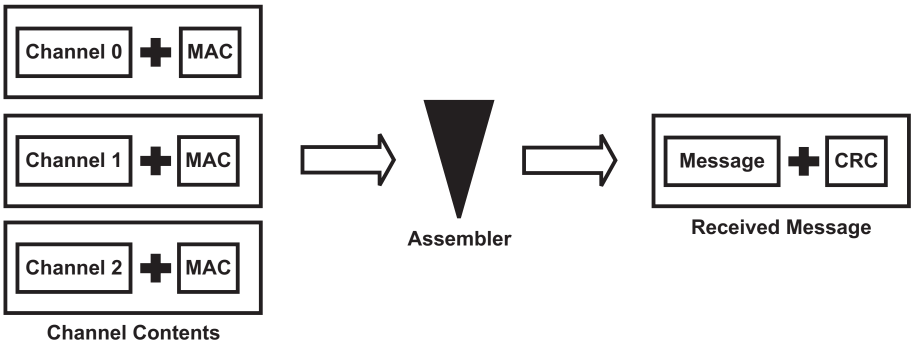

Figure 2. Three-channel message reception.

is reassembled to obtain the message and message CRC. The message CRC is used to verify the accurate transmission of the message.

3.3 Operation

The proposed framework has three phases of operation:

- **Initialization:** Alice and Bob predetermine the number of channels, number of fragments and fragment-channel mapping, and whether error detection or error correction is to be used. During a simulation, the modeler must also indicate the adversarial actions to be applied in the scenario. Before Alice can send a message, both Alice and Bob must compute the session key using the Diffie-Hellman Key Exchange algorithm. They each extract the most significant 160 bits of the shared key. The first 80 bits are used as the Trivium key and the next 80 bits are used as the Trivium initialization vector.

- **Sending:** In order to send a message, Alice assigns each individual bit of the message to a fragment based on the output of the Trivium cipher. After all the message (data and message CRC) bits have been assigned to fragments and the fragments have been assigned to one or more channels, appropriate error detection/correction bits are computed for each channel for transmission of its assigned fragments. If forward error correction is used, Alice computes a Reed-Solomon code for each channel output and appends the code instead of a channel CRC.

- **Receiving:** The receiver Bob begins by assuming that each channel is valid. If a channel is unavailable due to network degradation or adversarial action, the channel is marked as invalid. Bob then checks the contents of each channel using the CRC or Reed-Solomon code. If the channel CRC indicates an error or the channel Reed-Solomon code indicates an error that cannot be corrected, then the corresponding channel is marked as invalid. Otherwise, Bob recombines the message using the channel mapping. For a given bit, if there are differences between channels carrying the bit (i.e., Channel 1 and 2 both transmit a specific bit, but their contents differ), the particular fragment is flagged as having been modified. After the recombination process is completed, the message CRC is checked to see if it was received properly. In the event of a failure, the protocol goes into the recovery mode. If the modified fragments are isolated to a single channel, then a "smart recovery" is attempted by marking the channel carrying the fragments as invalid. If the smart recovery succeeds, Bob flags the channel modified by the adversary as being insecure.

Consider an example involving a three-channel communications session between Alice and Bob. Alice sends a message M, which is divided into three fragments, f_1, f_2 and f_3, with the following fragment-channel mapping:

$$\begin{aligned} C_1 &= \{f_1, f_2\} \\ C_2 &= \{f_2, f_3\} \\ C_3 &= \{f_3, f_1\} \end{aligned}$$

Eve conducts a man-in-the-middle attack on channel C_2 that changes f_2 to f_2' and f_3 to f_3'. This causes Bob to receive the following information on the three channels:

$$\begin{aligned} C_1 &= \{f_1, f_2\} \\ C_2 &= \{f_2', f_3'\} \\ C_3 &= \{f_3, f_1\} \end{aligned}$$

When Bob attempts to recombine the message, he detects differences between the two copies of both f_2 and f_3, indicating a potential adversary in channel C_2. He proceeds to accurately and correctly determine message M from only the contents of channels C_1 and C_3, and reports the adversarial presence in channel C_2.

If Bob was unable to isolate the modified channel, which could occur if Eve modified multiple channels, he could attempt to recover M by iterating through all the channel combinations that carried the entire message. This is possible because the message authentication code is spread across the channels, so even if a message in a compromised channel is accepted by Bob, the message authentication code would indicate the modification of the message. This recovery is computationally very expensive, but may be acceptable in some cases.

4. Insights

This section discusses the implementation challenges related to the proposed multi-channel framework, along with its complexity and effectiveness at mitigating attacks.

4.1 Implementation Challenges

The most glaring implementation challenge is to define the communications channels used by the sender and receiver. The challenge is addressed in this work by using a predefined channel set, but it limits the flexibility. Similarly, since the splitting mechanism relies on the communicated session parameters, the parameters must be transmitted via a key exchange mechanism over one or multiple channels in the channel set or be transmitted out of band. The transmission of session parameters is a design choice based on the security needs. Also, dynamic operation/channel configuration are required to address the possibility that a channel can become unresponsive or experience degraded performance. These challenges would likely be resolved as protocols such as Transport Layer Security are enhanced to support multi-channel systems.

Error detection and correction also pose challenges. Given a cryptographically secure splitting mechanism, if a bit is lost (and not recovered using forward error correction), then the joining mechanism would be unable to piece together the final message without knowing which bit was lost. Thus, error detection or correction must be implemented for every channel. The current solution relies on the message CRC and Reed-Solomon code for error detection and correction, respectively.

The other main challenge relates to security configuration. Relationships exist between the overhead of a security scheme and its resilience, confidentiality and integrity. The duplication of information increases the overhead while enhancing confidentiality and integrity. Confidentiality and integrity requirements do not need to be specified; instead, they are achieved by selecting an appropriate error correction mechanism and amount of duplication.

4.2 Time and Storage Complexity

The operation of the framework involves: (i) initialization and key exchange; (ii) message fragmentation and sending; (iii) message transmission; and (iv) receipt and recombination. The most computationally intensive part is the Diffie-Hellman key exchange. The security of key exchange relies on the discrete logarithm problem, which is discussed in [2]. Yakymenko et al. [13] have shown that the temporal complexity of modular exponentiation, which is required to compute a Diffie-Hellman key, is $O(b^2 \cdot \ln^2 b)$, where b is the size in bits of the modulus p used in the algorithm.

Fragmenting and sending a message M requires $\log_2(n)$ operations to generate a fragment for a message bit, where n is the number of channels. These operations must be performed m times, where m is the number of bits in message M. As each byte is processed, the error correction code for the message is computed, which requires $O(1)$ time. After fragmenting the entire message, an error correction code is computed for each channel, which requires $O(\frac{m}{8}) = O(m)$ time. Thus, fragmenting and sending a message requires $O(m \cdot \log_2(n))$ time. However, since the number of channels n is much less than the number of message bits m, the time complexity becomes $O(m)$. This also accounts for the inclusion of the message and channel error correction codes.

Message transmission is not included in the complexity analysis because it depends entirely on the transmission time and end-to-end delay of a communications link/interface.

Receiving and recombining a message operates similarly to message sending, with the exception that the system must recover from a message modified by an adversary. If each channel transmits securely, then message recombination requires $O(m \cdot \log_2(n)) = O(m)$ time. However, the recombination of a modified transmission depends on the allowed attempts. The system can attempt to recover from errors in x channels, where $0 \leq x \leq n - 1$ because there must be at least one correct transmission. Each recombination requires m operations and, in the worst case, it would require the full nested structure. Thus, the recombination would require $O(m^x)$ time.

The storage requirement depends primarily on the amount of duplication. The Trivium cipher was selected because its implementation requires minimal hardware, just 180 bits of state. System operation requires the ability to recombine a message from some of its parts (if there are errors), which means that the contents of each channel must be stored in a buffer. Because each channel could potentially carry the entire message, the buffers would require $\frac{m \cdot n}{8}$ bytes. Once again, because

$m >> n$, the storage requirement in practice would be $O(m)$, which is reasonable.

4.3 Attack Mitigation

As stated above, RFC 3552 clarifies the terminology for writing security considerations in future RFCs. Additionally, RFC 3552 specifies the following attack environment [11]:

> *"We assume that the attacker has nearly complete control of the communications channel over which the end-systems communicate. This means that the attacker can read any PDU (protocol data unit) on the network and undetectably remove, change, or inject forged packets onto the wire. This includes being able to generate packets that appear to be from a trusted machine. Thus, even if the end-system with which you wish to communicate is itself secure, the Internet environment provides no assurance that packets which claim to be from that system in fact are."*

This section discusses the effectiveness of the proposed framework in mitigating eavesdropping, jamming and man-in-the-middle attacks that target a single communications channel. As stated above, the attacker can access protocol data units in a channel if desired (except in the case of a jamming attack).

In an eavesdropping attack, an adversary only intercepts the portion of the message carried on the targeted channel. However, the adversary does not know which bits have been intercepted and how many bits are missing from the actual message. This significantly reduces the usable information obtained by the attacker.

A jamming attack seeks to undermine the availability of a targeted message. Let *df* denote the number of times each fragment is duplicated across the available channels. For example, if there are three channels, $df = 0$ means no duplication (each fragment is sent once), $df = 1$ means each fragment is sent twice and $df = 2$ means full duplication (each fragment is sent on all three channels). The adversary would certainly succeed if the duplication factor *df* is zero because the loss of a single channel would prevent message receipt. However, when $df \geq 1$, the receiver can recreate the message if no more than *df* channels are lost.

The most sophisticated man-in-the-middle attack involves an adversary who successfully modifies the information in a channel, including the channel CRC, so that the channel information is accepted at the destination. However, due to the recombination mechanism, the adversary only succeeds if more than *df* channels are modified. In addition, the receiver can determine which channel(s) have been affected while still receiving the message without retransmission. This mitigation is possible because the message CRC is interleaved in all the available channels.

In the implementation, no logical relationships exist between fragments in different channels. System resilience stems from the strict duplication of information and the dispersion of the message authentication code across the available channels. Methods exist for reducing the overhead involved in operating a multi-channel system while maintaining the same level of resilience. However, as the dependencies and relationships of the data in different channels increase, so does the amount of information that an adversary can gain. An example is a system with three channels C_1, C_2 and C_3, where each channel transmits equal length fragments. If $C_3 = C_1 \oplus C_2$, then even if an adversary compromises any one channel, the message can be recovered from the remaining two channels. However, a drawback of this approach is that all the channels are forced to carry information of equal lengths instead of probabilistically-equal lengths as implemented.

5. Discussion

This section discusses the implementation of the proposed framework in the TCP/IP architecture and in the critical infrastructure.

5.1 TCP/IP Implementation

A key issue is how the proposed framework would fit within the TCP/IP architecture. An argument could be made to include it as a session layer protocol because it relies on several channels, each of which would have its own TCP/UDP connection. However, the Transport Layer Security protocol was specifically designed to be flexible to accommodate a changing cyber security paradigm. Clearly, there is a need to develop multi-channel communications systems that provide security even if attacks outpace encryption schemes. Thus, there is a high likelihood that Transport Layer Security or another protocol would provide multi-channel security support.

The proposed system would work with a protocol that allows the creation and synchronization of multiple channels. Given a set of channels between a client and server, the framework would function as a cipher suite for the Transport Layer Security Protocol. Relying on the current Transport Layer Security nomenclature, it would merely be necessary to specify the method of key exchange as in the following examples:

- **TLS_RSA:** RSA.
- **TLS_DH:** Diffie-Hellman.
- **TLS_DHE:** Ephemeral Diffie-Hellman.

- **TLS_ECDH:** Elliptic curve Diffie-Hellman.

It is also necessary to specify the message authentication code to be used. Transport Layer Security currently utilizes hash-based message authentication codes (HMACs) with stream ciphers; these special message authentication codes provide message integrity and authenticity. They are currently denoted as follows:

- **HMAC-MD5**.
- **HMAC-SHA1**.
- **HMAC-SHA256/384**.

To match the Transport Layer Security nomenclature, the message authentication codes would be denoted as:

- **HMAC-CRC-32**.
- **HMAC-RS**.

This only leaves the session parameters for the fragmentation factor and duplication factor based on the number of available channels n. Thus, the client would offer the following items:

- **FF-X:** Number of message fragments ($X \leq n$).
- **DF-Y:** Session duplication factor ($Y < n$).

By modifying the proposed multi-channel system as a cipher suite for a future version of Transport Layer Security, the following session parameters would be communicated by the client at system initialization:

- **TLS_DH_CRC-32_FF-X_DF-Y**.
- **TLS_DH_RS_FF-X_DF-Y**.

5.2 Critical Infrastructure Implementation

The encryption and authentication methods utilized in the proposed framework would not immediately meet the security requirements for widespread implementation in critical infrastructure communications. However, the framework demonstrates several concepts that must be considered and addressed prior to an implementation. Also, it showcases the benefits of using multiple channels, especially when data is split at the bit level in a nonpredictable/pseudorandom manner. Even if an adversary could break the encryption used in a channel, the adversary would not know which bits have been decrypted and how the bits fit into the overall message. Thus, the adversary would have to intercept/compromise several channels to gain any meaningful information. This matches the confidentiality requirement specified in RFC 3552 for network communications [11]. Similarly, because the message authentication code is distributed across multiple channels, changes to a particular channel can be detected even if the adversary modifies the information so that it passes the channel-specific message authentication code. This matches the integrity requirement specified in RFC 3552 [11].

The proposed framework can thus be applied to existing critical infrastructure communications, where the primary concerns are message confidentiality and integrity. By fragmenting data across multiple channels for infrastructure communications, the adversarial actions needed to defeat the security mechanisms increase considerably. Specifically, multiple channels have to be intercepted and modified, and even if multiple channels are compromised, the adversary would still have to break the data splitting mechanism at the endpoints.

It is possible that the Trivium cipher may not provide adequate security for the splitting mechanism in some critical infrastructure scenarios. In such cases, the cipher may be replaced with a more secure alternative. Also, encryption can be applied to messages or individual channels or both, depending on the timing and overhead constraints.

6. Conclusions

Secure communications protocols should support multi-channel communications to leverage the security services that can be provided by multiple channels. The proposed multi-channel communications framework relies on data fragmentation in order to secure transmissions – it uses existing key exchange mechanisms to communicate initialization information, a splitting mechanism to map data to channels, and error detection and correction mechanisms. The framework also provides tunable parameters, namely the number of channels, duplication factor and number of fragments per message, which can accommodate user-specific security requirements. An important feature of the framework is its resilience to adversarial actions, including eavesdropping, jamming and man-in-the-middle attacks. For example, the framework can detect and defeat man-in-the-middle attacks without retransmission while reporting the channels that were compromised. However, some challenges need to be resolved prior to implementation, including securely determining the channel set prior to initializing communications sessions.

The framework can serve as a roadmap for developing secure, multi-channel communications systems because it demonstrates what is necessary to meet key security requirements and illustrates the challenges that must be addressed before implementation. The framework would dovetail nicely with future implementations of Transport Layer Security and other protocols. The resulting multi-channel communications system could be tailored to user needs, gaining corresponding increases in confidentiality, integrity and availability even in the presence of adversarial actions. Indeed, the multi-channel system would significantly increase the overhead required by attackers to gain meaningful information (confidentiality), modify transmitted information (integrity) and prevent information from being used (availability).

The views expressed in this chapter are those of the authors, and do not reflect the official policy or position of the U.S. Air Force, U.S. Department of Defense or U.S. Government. This document has been approved for public release, distribution unlimited (Case #88ABW-2019-6022).

References

[1] R. Barnes, M. Thomson, A. Pironti and A. Langley, Deprecating Secure Sockets Layer Version 3.0, RFC 7568, 2015.

[2] I. Blake and T. Garefalakis, On the complexity of the discrete logarithm and Diffie-Hellman problems, *Journal of Complexity*, vol. 20(2-3), pp. 148–170, 2004.

[3] C. De Canniere and B. Preneel, Trivium Specifications, Computer Security and Industrial Cryptography Group, Department of Electrical Engineering, Catholic University of Leuven, Heverlee, Belgium (`www.ecrypt.eu.org/stream/p3ciphers/trivium/trivium_p3.pdf`), 2006.

[4] T. Dierks and C. Allen, The TLS Protocol Version 1.0, RFC 2246, 1999.

[5] T. Dierks and E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.1, RFC 4346, 2006.

[6] T. Dierks and E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.2, RFC 5246, 2008.

[7] S. Fries and R. Falk, Ensuring secure communications in critical infrastructures, *Proceedings of the Sixth International Conference on Smart Grids, Green Communications and IT Energy-Aware Technologies*, pp. 15–20, 2016.

[8] North American Electric Reliability Corporation, United States Mandatory Standards Subject to Enforcement, Atlanta, Georgia (`www.nerc.com/pa/stand/Pages/ReliabilityStandardsUnitedStates.aspx`), 2020.

[9] C. Paar and J. Pelzl, *Understanding Cryptography: A Textbook for Students and Practitioners*, Springer, Berlin Heidelberg, Germany, 2010.

[10] E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.3, RFC 8446, 2018.

[11] E. Rescorla and B. Korver, Guidelines for Writing RFC Text on Security Considerations, RFC 3552, 2003.

[12] C. Wolfe, S. Graham, R. Mills, S. Nykl and P. Simon, Securing data in power-limited sensor networks using two-channel communications, in *Critical Infrastructure Protection XII*, J. Staggs and S. Shenoi (Eds.), Springer, Cham, Switzerland, pp. 81–90, 2018.

[13] I. Yakymenko, M. Kasianchuk, S. Ivasiev, A. Melnyk and Y. Nykolaichuk, Realization of RSA cryptographic algorithm based on vector-module method of modular exponentiation, *Proceedings of the Fourteenth IEEE International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering*, pp. 550–554, 2018.