Intelligent Cipher Transfer Object for IoT Data Security

Bishal Sharma
Ingram School of Engineering
Texas State University
San Marcos, TX, USA
email: dxa6@txstate.edu

Bishal Thapa
Ingram School of Engineering
Texas State University
San Marcos, TX, USA
email: b_t220@txstate.edu

Stan McClellan
Ingram School of Engineering
Texas State University
San Marcos, TX, USA
email: stan.mcclellan@txstate.edu

Abstract—**The availability of robust data security technologies to provide end-to-end, verifiable provenance of information is increasingly important. This study explores the new Intelligent Cipher Transfer Object (ICTO) technology as a novel approach to comprehensively securing digital data. The technology is assessed in terms of its performance and robustness relative to current security and data transport paradigms. Machine Learning algorithms are used to identify residual artifacts that may be exploited, and potential security threats associated with ICTO are described.**

Keywords-ICTO; IoT; Cybersecurity; Information Security; Blockchain; TLS; Encryption; Machine Learning; Cryptanalysis.

I. INTRODUCTION

This paper compares and contrasts key performance characteristics of technologies which are commonly used for securing information, both at rest and in transit, with emerging Intelligent Cipher Transfer Object (ICTO) technology. In previous work, we compared the performance of ICTO with well-known network transport technologies such as MQTT and conventional TCP [1]. The present work is an extension of the previous evaluation to a broader set of technologies and performance metrics, including the use of Machine Learning to evaluate the properties and potential weaknesses of a commercially available ICTO implementation. We conclude that ICTO may offer comprehensive data security independent of data location or transport even though aspects of data leakage may prove complex to resolve.

The current state-of-the-art in data security focuses on securing data when it is traveling (in transit) between network endpoints. Several complex operations, including payload encryption, may be performed on the data so that it cannot be accessed or modified during transit. However, when data is not in transit (at rest), it may be in possession of users, applications or systems where it may not be protected. Generally, multiple techniques or mechanisms are required in modern commercial or public data exchange settings (e.g., any client-server based interchange or web service) to securely transfer a piece of information from one point to another. These mechanisms have to be tightly integrated with one another to prevent any data-related leakage or other accidental disclosure of private information. The novel ICTO technology addresses the issue of robust integration of complex security techniques by creating a secure “intelligent”, “self-aware”, and “self-governing” object that can allow, deny, track, lock, or destroy itself based on the entity that is trying to access it, regardless of the security posture of the public (transitory) communication channel, or private (resting) environment. However, such a holistic approach to secure information may be costly in terms of computational or network performance.

After brief introductions to the specifics of blockchain and ICTO in Section II, comparative performance analyses are provided in Sections III-VI which contrast ICTO with several conventional approaches to information security. Section III describes the general experimental setup and important parameters. Section IV presents system-level performance measures which were collected and analyzed. Section V compares blockchain and ICTO in terms of computational performance and storage requirements, considering identical payloads. Section VI analyzes an ICTO implementation using cryptanalysis and unsupervised machine learning (ML). Section VII summarizes the experimental results and provides useful conclusions for the various technologies and implementations. Section VIII summarizes potential future work with these technologies.

II. BACKGROUND

Data or information security is the science of using and developing tools and techniques to prevent unwanted access to information. The fundamental elements of data security include:

- Confidentiality, which protects information from unauthorized access,
- Integrity, which guarantees accuracy and completeness of data providing assurance that it has not be tampered with,
- Availability, which makes information available to authorized parties whenever necessary,
- Authentication which provides a means of verifying the identity of users,
- Authorization, which grants access to specific resources to a user’s identity, and
- Non-repudiation, which takes away false denial of possession or origination of information.

These key elements are necessary to provide individuals and organizations with assurance that their privacy is maintained and information they are using is trustworthy and invulnerable to threats, regardless of how such data may have been generated.

Cybersecurity systems are designed and implemented using some combination of hardware and software so that these fundamental characteristics are in place. In an interconnected computing framework, data resides either inside memory within server systems or an end-client computer – considered to be "at rest", or in a network channel traveling from one endpoint to another – considered to be "in transit." Organizations may use public cloud infrastructures or private computing infrastructures to manage and store data [2].

Cloud computing technology is an internet based, decentralized, distributed, and virtualized computing paradigm in which operations on data as well as data transportation is carried out, often through web-based services [3]. This technology offers scalability, ease of deployment, and cost- effectiveness among other benefits [4]. However, from the perspective of cybersecurity, there are some issues and challenges that require extra preventive measures by both the cloud service provider as well as the cloud service user [5-6]. Although a typical cloud security stack includes fundamental services like authentication, access control, and encryption, data loss may still be encountered due to a plethora of issues, including server or application misconfiguration, malicious attacks, insecure data-flow pipelines, vulnerable Application Programming Interfaces (API), and mislocated data [7].

Legacy systems, which typically use privately owned/leased, on-premises systems to process digital data may seem to provide better security as data resides within relatively secure boundaries. However, complex logistical factors provide motivation to migrate toward cloud platforms, including high setup and operating cost, reduced flexibility, complicated integration, deployment, and maintenance procedures [8].

Considering the various modalities for data generation, transfer, and storage, employing the convenient perspectives of "application security", and "network security" is warranted [9]. For instance, encryption is a mechanism that is designed to prevent access to data travelling in a network. So, encryption can be regarded as a "network security" technique. However, authentication mechanisms – more broadly – those similar to Role Based Access Control (RBAC) are used to allow data access to known users, and so is an "application security" mechanism. Conventional Authentication, Authorization, and Accounting (AAA) security frameworks that enforce access controls to data and network resources and maintain accountability of network resources are prime examples of network security that is widely accepted and implemented. A robust cybersecurity framework should provide both "application security" and "network security" components to ensure protection. The Internet of Things (IoT) is an example of an important and complex domain where data security concepts may need to be viewed using multiple perspectives.

A. *Security Frameworks*

To gain a better idea of security of data in transit (network security) and at rest (application-level security) in modern frameworks, we use the well-known "Alice and Bob" scenario where users are communicating over the internet using a messaging website. Alice wants to say "hello" to Bob, so she uses her browser to access the messaging website. To establish a secure link, Alice's computer and the website server use public-key cryptographic schemes to first authenticate each other, and then set up a symmetric session key to encrypt outgoing messages from Alice's browser. The use of encryption along with the established session key secure the channel between the website's server and Alice. Several complex algorithms including Rivest Shamir Adleman (RSA), Diffie-Hellman, Secure Hash Algorithm (SHA), digital signatures, digital certificates, and Advanced Encryption Standard (AES) are used just to get to this point [10]. As soon as Alice's data reaches the server, it gets decrypted and thus is no longer obscured. Since the messaging website would most likely be hosted by an enterprise that handles user data, it uses IAM mechanisms to make sure that such data cannot be accessed by unauthorized or unintended entities (applications or individuals) within the enterprise. This can be seen as protecting data at rest [11]. A similar procedure is followed between the server and Bob as he receives Alice's message.

This example provides a simplified but useful perspective on modern security frameworks which employ channel security and site security for data in transit and at rest. It is important to note how the several complex cryptographic operations and access-control systems are necessary to ensure end-to-end data protection. Successful attacks on such systems are commonplace. An example of an in-transit data breach includes the 2009 attack on the A5/1 encryption algorithm used in 2nd generation Global System for Mobile Communication (GSM) network [12]. An example of an at-rest data breach includes a leak of the personal data of 100 million Capital One customers due to misconfiguration of a web application firewall in the company's cloud infrastructure [13]. Such attacks (among many, many others) show that complex integrations are prone to misconfigurations or mismanagement that can lead to catastrophic results.

Security frameworks in IoT networks use similar integrations. However, authentication and encryption of IoT sensor data is usually performed by a network gateway or other intermediate system instead of by an end-device. This architecture allows for exploitation of potentially vulnerable or unprotected links between the sensors, end-devices, and gateway or intermediate system.

Many alternative concepts have been developed to address issues in current security frameworks, including secured data self-moderating access and authorization, inserting encryption/decryption algorithm as metadata into data files, and specifying access and authorization criteria into the data objects [14-15].

B. Blockchain

A notable technology gaining traction for IoT-related applications is Blockchain. Blockchain is an eponymous, immutable ledger that stores transactions in blocks of data which are linked together like a chain. Blockchain can also be defined as a decentralized, shared, immutable database that makes it easier to track assets and record transactions in a network [16], enabling transparent information sharing.

The blocks in a blockchain are linked together by a hash function. Every block has a timestamp, transaction, and the hash of the previous block. The first block in the blockchain is referred to as the "Genesis block". The embedded hash validates the integrity and non-repudiation property of the data that is stored inside the block [8]. Every participating node in the blockchain network is updated regularly with the copy of the original. On a blockchain network used as a distributed ledger, practically anything of value (e.g. cryptocurrency) may be recorded and traded, lowering the risks involved for all parties [16].

Unfortunately, sophisticated security techniques such as blockchain require greater processing resources and power. As a result, blockchains have been modified to address the resource constraints in various domains. IoT devices typically do not have powerful processors and large memory. Instead, they possess just enough computational resources to periodically perform a limited number of tasks. Since these devices are battery operated, relatively inexpensive, and designed for highly specific purposes, they are not usually equipped with substantial data protection mechanisms. Techniques designed for general purpose, commercial, or enterprise systems therefore may not be well suited for application in the IoT space.

Often, the term "blockchain" is understood in casual usage to mean "cryptocurrency" due to the recent popularity of digital currency exchanges. Blockchain is the core technology behind cryptocurrency, but the application of blockchain is not limited to cryptocurrency. The distributed ledger is the major technology that introduced the concept of a Peer-to-Peer Electronic Cash System called Bitcoin in 2009 [17]. Distributed ledgers do not rely on centralized governance and exist virtually or digitally. This legitimate use of distributed ledgers, and the cryptography for securing transactions has made it a reliable alternative to traditional banking systems. The crypto architecture leverages computational power to solve complex mathematical puzzles, a process commonly known as "mining" [18]. Mining yields cryptocurrency units such as Bitcoin or Ethereum. While the influence of blockchain on IoT may seem obvious, a debate of "where to host the blockchain" remains [19]. Several implementations of blockchain technology exist to suit different applications.

Distributed ledgers address the "single point of failure" issue. Consensus mechanisms such as Proof of Work (PoW), Proof of Authority (PoA), Proof of Stake (PoS), Delegated Proof of Stake (DPoS), and Practical Byzantine Fault Tolerance (PBFT) provide a trustworthy distributed system [20]. The most common consensus mechanism is PoW which relies on computational work to validate new blocks and add them to the distributed chain. In contrast, systems implementing a centralized architecture [21] may be less reliable as sensitive data is consolidated, making it a prime target for cyberattacks [21]. The risk of tampering and counterfeiting is addressed in blockchain by the distributed nature of the ledger. Multiple sources confirm the transactions before recording. This provides high levels of reliability and security. Some applications manipulate a tremendous volume of data, and it isn't practical to store all the data in the blockchain itself. In such cases, decentralized storage or off-chain storage can be used.

Regardless of architecture or application, distributed ledgers and Blockchain can be effective in certain application domains and can provide a practical framework for distributed data storage as well as protection [22].

C. Intelligent Cipher Transport Object (ICTO)

Protecting information is difficult when all design parameters are considered, especially for IoT. Even if the channel is assumed to be secure, securing endpoints is still a challenge. A new technology capable of securely encapsulating data and embedding it with thorough access control policies may be a promising approach to address security issues in modern systems, irrespective of user, device, network, or operating system. This research aims to explore the usefulness of one such specific technology in terms of security and efficiency - Intelligent Cipher Transfer Object (ICTO) [23].

ICTO is a security technology that includes mechanisms for participant authentication and authorization for access of data which is protected by cloaking patterns. A portable dynamic rule set, which includes executable code for managing access to the protected set of participants and the protected data, is included within the ICTO. For a given user, the ICTO may provide access to some participants while preventing access to other participants based on this set of constraints [23]. The ICTO concept extends the idea of conventional AAA and RBAC concepts by cloaking data at the point of generation with specific user-defined rule sets. The owner of the data has substantial control over how or when protected data can be accessed by another party, regardless of the storage, transport, or operational environment.

ICTO is a form of encapsulation which achieves data security by embedding security techniques into the data itself. This provides a form of self-defense, user authentication, and governance and tracking ability which is independent of network or system security [23]. The key feature of this concept is that it allows data to be protected at the point of origin, eliminating a dependence on the security of the communications channel. In addition, access control policies and authentication parameters can be set by the data owner during object creation and therefore eliminate the need for third-party digital certificate providers. Fig. 1 shows the components/modules that are embedded with user data to create an ICTO object, also called "digital mixture" or "self-governing data" [23]. Implicit in the use of ICTO is a common execution platform or trusted set of libraries implemented on systems which manipulate ICTO objects.

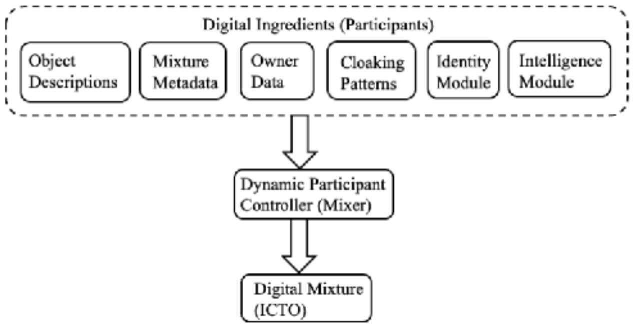


Figure 1. ICTO Creation Process.

The ICTO comprises a set of participants including a portable dynamic rule set (PDRS) which is responsible for enforcing/evaluating the rules and policies in order to allow/deny access to an entity attempting to access data. Fig. 2 and Fig. 3 illustrate simplified use cases where users can protect their data and can define specific conditions to allow or deny access depending on policies/rules set during ICTO object creation.

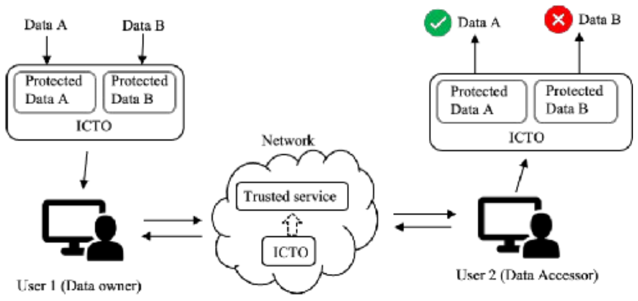


Figure 2. ICTO use case. ICTO object created by User 1 for partial data access by User 2 can be transported over an insecure network [23].

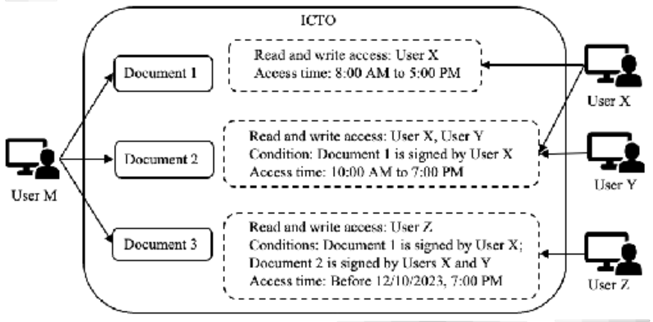

ICTO
Document 1
Read and write access: User X
Access time: 8:00 AM to 5:00 PM
Document 2
Read and write access: User X, User Y
Condition: Document 1 is signed by User X
Access time: 10:00 AM to 7:00 PM
Document 3
Read and write access: User Z
Conditions: Document 1 is signed by User X;
Document 2 is signed by Users X and Y
Access time: Before 12/10/2023, 7:00 PM
User M
User X
User Y
User Z

Figure 3. Access control policies in an ICTO object. Example of multiple access control policies configured within an ICTO object created by user M for multiple users X, Y, and Z [23].

Through software and tools, cipher objects containing cloaked data along with other modules are created that can only be utilized/deciphered by using compatible software.

The necessary software and tools used for this study was provided by Sertainty Corporation. Sertainty's UXP is an implementation of the ICTO concept that focuses on protection at the data layer, targeting any kind of unstructured/structured data format. UXP objects are essentially a secure, portable filesystem that hides data and access policies within a single file. For simplicity, the terms UXP objects, UXP files, ICTO, or ICTO objects are used interchangeably in this paper.

To create a UXP object, an XML file specifying user identification and definitions, access control and authorization policies, and other information is required. This XML file is used to generate a protected ID file that is then used during creation of the UXP object. The ID file is a digital object containing access control and authorization policy information, user definitions, authentications parameters etc. in a secure format. The ID object and user data are combined to create the UXP object (".UXP" filename extension). Fig. 4 summarizes the process of combining the XML file containing data policies, the resulting ID object (".iic"), and the provided user data to create a UXP object, which is an instance of an ICTO.

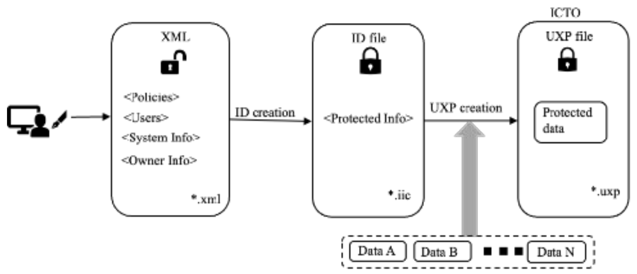


Figure 4. ICTO object creation process.

III. Experimental Setup

The ICTO implementation considered in this research is proprietary, and the property of Sertainty Corporation. The Sertainty UXP technology is chosen for evaluation as a compelling instantiation of the ICTO concept.The purpose of the outcomes in this paper summarize our understanding of the security features and performance of this promising technology in comparison with other similar competing or enabling technologies. Most experimental results are phrased in the context of an IoT-related application, and the technologies are compared or contrasted from this perspective.

A. System Performance Measurement

The computational cost of using an ICTO to secure user data is a concern, particularly for resource-limited systems. Thus, this work investigated parameters associated with ICTO creation by gathering system resource usage/overhead and network overhead data related to creation of UXP objects.

To gather performance data, a computer running a Linux operating system was used. The computer system was equipped with an Intel i5 processor having an average clock speed of 2.4 GHz, 5 MB cache, and 8 GB RAM. Linux shell and Python programming languages were used for running various experiments and for data processing.

To gather data related to network performance, secure and unsecure user data in plain text format was transported via both secure and unsecure channels to another machine with similar specifications in a private network (LAN) setting.

Statistical evaluation of conventional system burden including clock cycles, memory usage, and elapsed time provided a baseline for system performance analysis whereas metrics including total transmit time, transmission overhead, and related network parameters provided a baseline for network performance analysis.

All experiments were repeated at least 200 times for each independent variable (user payload size) to obtain statistically meaningful interpretation of the resulting data. Mean and 95% confidence intervals were calculated for the statistical study, and maximum/minimum values were noted.

For experimental data related to memory utilization, the following memory metrics were monitored:

- Data resident set size (DRSS) – The amount of main memory occupied by the data segment of a running process, excluding code/instruction memory.
- Proportional share size (PSS) – The portion of main memory occupied by a process, composed by the private memory of that process plus the proportion of shared memory with one or more other processes.
- Resident set size (RSS) – Represents the total amount physical memory (RAM) currently occupied by a process, including memory for both executable instructions and data.
- Virtual set size (VSZ) – A measure of all memory that a process can access, including memory that is swapped out, unused allocated memory, and memory used by shared libraries.
- Number of bytes which a task causes to be read from storage.
- Text resident set size (TRS) – The amount of memory devoted to executable code.

These memory metrics cannot be used as a reliable estimate for system memory utilization when used individually. However, coherent results can be gathered if all of these metrics are considered.

B. *Network Performance Measurement*

Network performance is typically measured by looking at parameters including total transmit time, control plane overhead, round trip time of packets, and similar. For analyzing the network performance of the ICTO technology, two important metrics of total transmit time and control plane overhead were recorded and statistically analyzed.

The experimental setup involved two client and server computers with similar specifications. The client created user data that is protected via the UXP implementation of ICTO [23], and performance of the client system during UXP creation and transport was recorded. For comparison, system performance was measured in three configurations:

1. Unprotected user data sent via "plain" TCP.
2. User data protected using UXP and sent via TCP.
3. User data sent via TLS over TCP.

Since TLS v1.2 and TLS v1.3 are the most common protocol for secure communication over the Internet, TLS v1.3 was selected for the experiments [24]. In addition to comparative performance measurements, experiments were also performed to observe how compressible and incompressible data types are handled during UXP encapsulation. Plain text containing ASCII characters and digits, as well as image files of the same fixed sizes comprised the user payloads for UXP objects.

Linux utility 'forkstat' [25] was used to capture process IDs (PIDs) of processes for Python-based programs on the client system and supplied to the 'perf' utility [26] for CPU monitoring as well as to the 'ps' utility for memory monitoring. The remaining test configurations were identical to those used for system performance assessment discussed in Section III.A.

C. *Blockchain Comparison*

Blockchain implementations are often customized based on use cases where the objective is to solve some critical issue. This research uses a version of blockchain with a centralized cloud architecture optimized for lightweight endpoints. This implementation addresses complexity and scalability issues related to the traditional distributed ledger. However, this model doesn't address the "single point of failure" that is solved by distributed ledger.

For experimentation purposes, an alternative version of blockchain was also designed as an implementation of a distributed ledger. This implementation incorporated the PoW consensus mechanism with four difficulty levels. PoW is a popular blockchain consensus mechanism widely used in distributed ledger technology. Thus, the blockchain architecture incorporating the PoW consensus mechanism is referred to here as the "ledger." The experimental setup involved running the blockchain and ledger on a Linux system with 8 GB RAM and 4 CPUs operating at 2.4GHz.

The payloads for blockchain and ledger experiments were compressed (for losslessly compressible payloads) using the LZ77 algorithm with Huffman coding. Compressible and incompressible payloads were also encrypted using Elliptical Curve Cryptography (ECC) hybrid encryption with Advanced Encryption Standard (AES). Python's 'tinyec' library was used to generate an ECC key pair [27].

IV. System Level Performance Measurements

This section presents experimental results pertaining to system related as well as network related parameters such as CPU time, main memory, elapsed user time, and total transport time. The results are compared with other settings for a more comprehensive assessment.

A. *System Performance*

System performance measurement includes two primary phases of experiments. In the first phase, system statistics are recorded during UXP object creation. The second phase compares to the size of the payload data with the size of the resultant UXP objects.

To measure first-phase system performance, UXP objects were created with varying payload (user data) sizes of 1 byte, 1 kilobyte (kB), 20 kB, 100 kB, 500 kB, 1 megabyte (MB), 2 MB, and 3 MB. Performance metrics including number of CPU cycles, memory allocated, and total elapsed time for creating UXP objects are presented in Fig. 5 and Fig. 6 along with 95% confidence intervals, computed using the t-distribution for data with unknown variance.

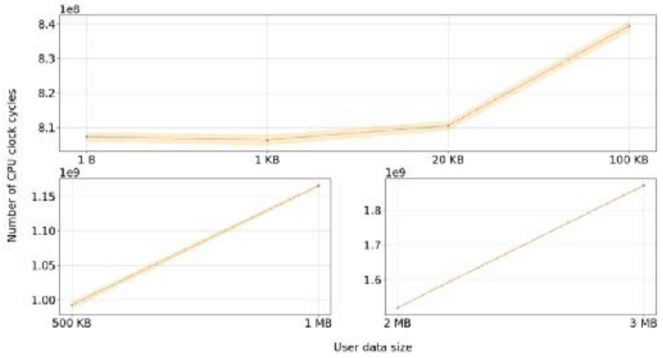


Figure 5. Plots showing number of clock cycles necessary to create UXP objects. User data size (X-axis) represents the size of user data (payload) that is protected by the UXP object. The lower set of plots accentuate the 95% confidence intervals, and indicate consistent, limited variability in the UXP creation process.

Fig. 5 shows that as the size of the UXP payload increases beyond 100 kB, the number of CPU cycles also increases, which is logical. For the smallest payload (1 byte), around 0.8 billion cycles are required for the minimum number of clock cycles to create a UXP object. As payload size increases, the number of CPU cycles increases steadily. The number of cycles required for the maximum payload size is twice the number of cycles required for the minimum payload size.

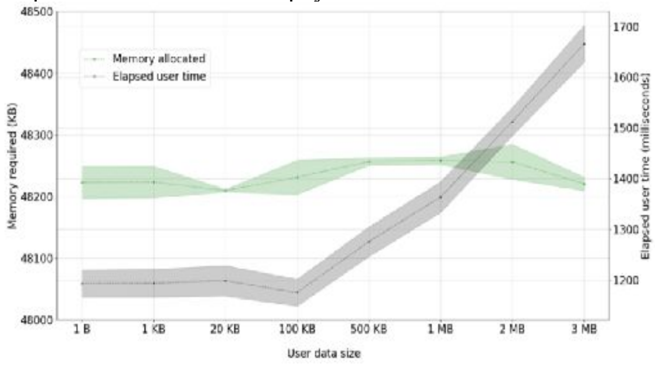


Figure 6. 95% confidence interval plot of memory utilized/time spent during UXP creation vs. size of user data protected. System memory (RAM) used is shown in the left side Y-axis and time spent is shown in the right-side Y-axis.

Fig. 6 contains two sets of data: allocated memory and elapsed time. The first set of data plotted in Fig. 6 shows that for payloads of 1 B to 100 kB, elapsed time remains relatively constant. Thus, a minimum time-to-create Fig. 6 of roughly 1 second may be observed which will vary based on system characteristics.

The second set of data plotted in Fig. 6 suggests that the total memory allocated/utilized during UXP creation remains fairly constant at under 48.3 MB regardless of payload size.

Thus, on average a minimum of 1 second and 800k cycles are necessary to create a useable, secure UXP object, based on an allocated memory of just under 48.3MB. These performance figures increase essentially linearly with payload size beyond 100kB (elapsed time) and beyond 20kB (cycles). As a result, a payload of 4MB would be expected to require approximately 1.9 billion cycles, would consume roughly 48.3MB, and would complete in under 1.8 seconds on a system comparable to those used for testing.

UXP object creation produces different results for different types of user payloads. Fig. 7 shows the size of UXP files after protecting image and text files of varying sizes. For text data, the resulting UXP object is smaller than the data for payloads larger than about 2500 kB.

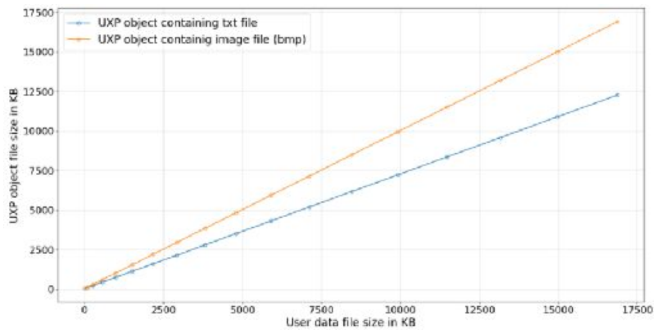


Figure 7. UXP object size for varying sizes of text and image payloads.

When the user payload is comprised of text data, the difference in resultant UXP file and original data file is negative for payloads around 264 kB. This difference increases with payload size suggesting that a lossless compression mechanism is employed during UXP object creation. However, with incompressible payloads, an almost constant positive difference is present regardless of payload.

B. Network Performance

After measuring system performance for UXP object creation, the objects were transported via a controlled network and network performance was observed. Network related performance was analyzed using system metrics such as RAM and CPU usage as well as transmission metrics such as control plane overhead and transmit time.

1) System Performance During Transport: To evaluate system performance, the number of CPU cycles and total memory allocated when unprotected user data is transported using "plain" TCP and using a TLS protected TCP channel is presented. These results are contrasted with similar results from the transport of UXP objects using "plain" TCP. Processing and memory requirements during transport of payloads of multiple sizes, across multiple settings are summarized in Fig. 8 and Fig. 9 using mean values with 95% confidence intervals.

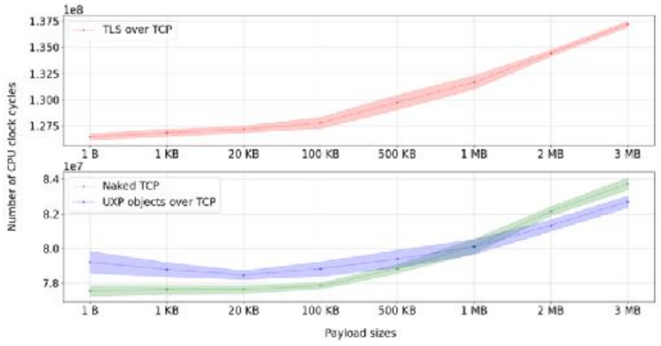


Figure 8. 95% confidence interval plot of number of CPU cycles for various payload sizes during transport.

The upper portion of Fig. 8 shows the number of CPU cycles vs. user payload with a 95% confidence interval when user data is transported via TCP protected with TLS. The lower portion of Fig. 8 shows the same data for plain TCP (user data unprotected), and for plain TCP with user data protected via UXP. From the plots, it is clear that using TLS for security is about 1.6 times more computationally expensive than using UXP objects as the security medium over a plain TCP channel.

As expected, CPU cycles increase steadily as the user data size increases. Surprisingly, the number of CPU cycles required to transport UXP objects via TCP gradually decreases as payload increases as compared to transport of unprotected data over TCP. This may be explained using Fig. 8, which suggests that UXP object creation includes lossless compression of user data.

Regardless, this observation suggests that protecting user data using ICTO technology provides improved performance for data larger than 500 kB.

Thus, UXP is substantially more efficient than TLS in channel and CPU utilization as well as transmit time, particularly for large payloads, and provides benefits for data at-rest as well as in-transit.

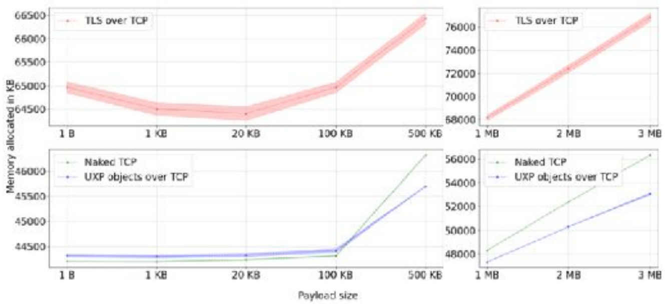


Figure 9. 95% confidence interval plot of amount of memory allocated during transport vs. user data size.

Fig. 9 shows allocated memory vs. user data sizes during transport of user data over plain TCP, over TLS secured TCP, and UXP-secured user data over TCP. All plots employ 95% CI.

As shown in Fig. 9, transporting user data over TLS is the most expensive in terms of allocated/required memory as well as compared to the other two modes of transport. TLS secured data transport is observed to generally require an additional 20 MB of memory compared to unprotected data transport or ICTO protected data transport.

Further, the memory required seems to be lower when UXP objects protecting user data larger than 100 kB are transported over TCP. Contrary to intuitive expectation, UXP protected data transport over TCP is observed to be more memory efficient than transport of raw unprotected data over TCP.

2) Network Performance During Transport: For analyzing the network performance of the UXP technology, two important metrics – total transmit time, and control plane overhead were recorded and statistically analyzed.

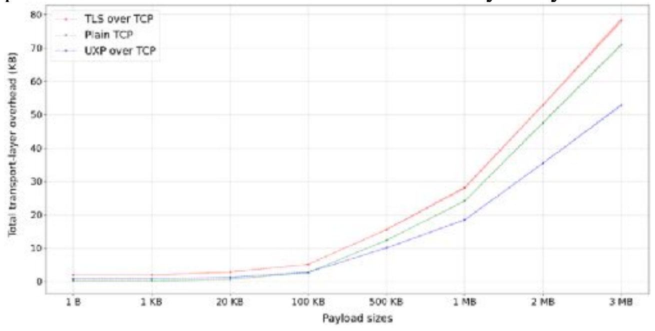


Figure 10. 95% C.I. plots showing control plane overhead vs. user data size. Overhead measurements are from the transport layer and above.

Mean and 95% confidence interval figures of total control plane overhead size for transmission as well as total transmit time for various payload sizes in multiple settings are presented in Fig. 10 and Fig. 11.Fig. 10 indicates that using UXP objects as a means of transporting user data seems to be most efficient in terms of overhead because it has the least amount of control plane overhead during transmission. Using TLS over TCP for secure data transport has at least 50% larger overhead as compared to secure transport of user data using UXP objects. On average, about 1950 bytes of overhead is introduced by TLS over TCP for transporting a single byte of data whereas UXP object and raw data transport over plain TCP introduce only 862 and 184 bytes of overhead respectively, which increases with payload size.

Hence, for transporting a single byte of data, using UXP objects as a means of securing the payload introduces only 50% of the overhead of TLS. For the maximum payload size (3 MB), TLS over TCP requires overhead of about 78 kB whereas using UXP objects over TCP requires 52 kB. Thus, for transporting 3 MB of user data, TLS introduces 1.5 times more network overhead. Further, compressible payloads reduce the size of UXP objects, so that required network overhead is less than 50% as compared with plain TCP. Thus, UXP is substantially more efficient than TLS in channel utilization, particularly for large payloads, and provides benefits for data at-rest as well as in-transit.

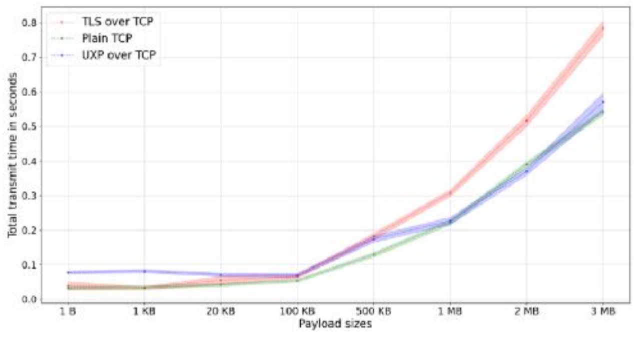


Figure 11. 95% C.I. plots of transmit time vs. payload size. Transmit times are calculated for transporting user data over TCP, user data protected with UXP objects over TCP, and via secure channel using TLS over TCP.

Fig. 11 shows the average transmit times for varying payload sizes for transport of user data in different configurations. As shown in the Fig. 11, the total time for transmission remains below 100 milliseconds for payload sizes up to 100 kB. Time required to transmit 1 byte to 100 kB payloads are highest when transporting UXP objects over TCP and the lowest when transporting raw user data over TCP. However, for payloads larger than 500 kB, UXP protected user data requires less time to transport than other configurations. The plot shows that for higher user payload sizes, transmission time is significantly lower for UXP object transmission over TCP than for that for TLS over TCP. Thus, UXP is substantially more efficient than TLS in transmit time, particularly for large payloads, and provides benefits for data at-rest as well as in-transit.

Although the transmit-time data presented in Fig. 11 were recorded in a controlled network environment, some incoherency is still evident. This may be explained by irregular handling of traffic by the network access point that connects the client and server machines. In addition, the interfaces at the communicating machines (client and server) may also have irregular scheduling/process priority for certain network related processes, affected by services/applications running in the background. Nonetheless, somewhat distinct trends are still observable and are enough to draw meaningful conclusions.

V. Blockchain vs ICTO

The results of experiments and data gathering consists of experiments performed on a general-purpose computer running the Linux operating system. The experiments were further categorized into two parts: compressible data, which encompasses experiments conducted with lossless compression techniques on payloads, and incompressible data, which comprises of payloads that could not be compressed using lossless compression techniques.

A. *Storage Comparisons*

Memory storage is important irrespective of the context of data security applications, and in UXP as well as blockchain implementations. In this section, a general comparison between blockchain technology and UXP is presented, considering that both versions of the blockchain i.e., core blockchain and ledger have the same memory requirements for the experiments.

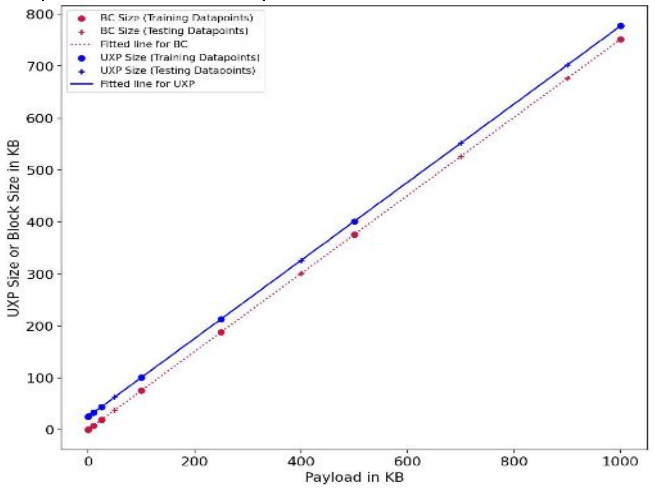


Figure 12. Fitted line for blockchain and UXP storing compressible payload with training and testing datasets.

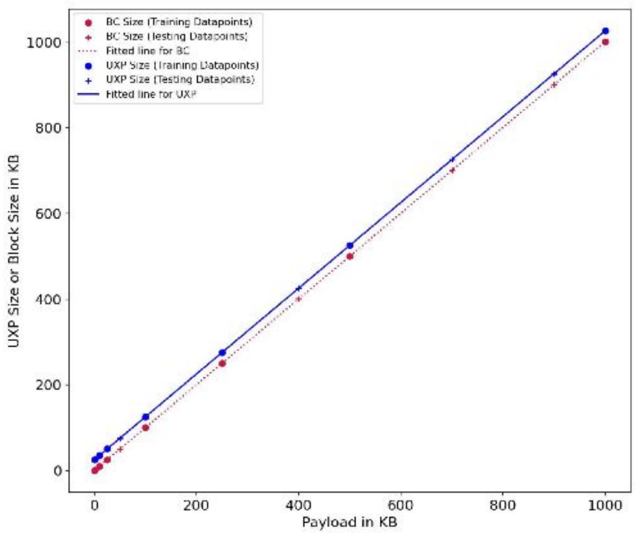


Figure 13. Fitted line for blockchain and UXP storing incompressible payload with training and testing datasets.

Fig. 12 and Fig. 13 summarize experimental analysis regarding the memory storage used by blockchain and UXP for compressible and incompressible payloads. The experimental dataset was used to perform linear curve fitting and generate equations that serve as an approximation for determining the block size and UXP size for various payloads. The relationship between the size of the payload and size of the protected payload was found to be linear.

Payload sizes within specific range of 1Byte to 1MB was used as the training dataset for fitting the linear equations. To assess the accuracy and performance of the equations, a separate testing dataset was prepared. The testing dataset included payload sizes of 25kB, 400kB, 700kB and 900kB for both experiments. It is evident from Fig. 12 and Fig. 13 that the testing datapoints, represented by red dots for blockchain and blue dots for UXP, align closely with the straight line approximated by the training dataset. The confidence interval had a significantly low range, which made it infeasible to illustrate in Fig. 12.

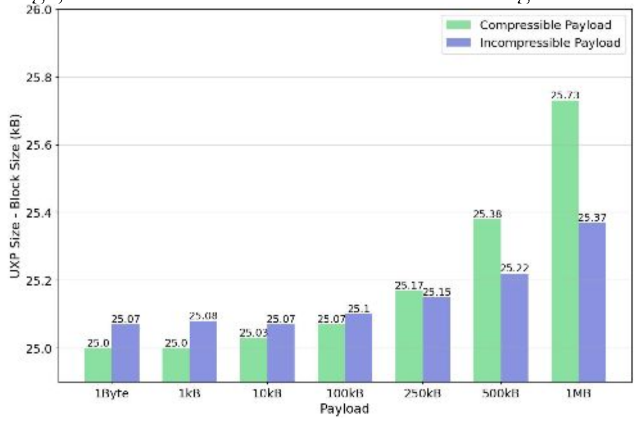


Figure 14. Difference in UXP Size vs Block Size.

Fig. 14 displays the disparity in size between a block and UXP encapsulating compressible and incompressible payloads of the same size. Storing the same size payload in a UXP object vs. a blockchain block requires a mean of 25.20 kB for a compressible payload and 25.15 kB for an incompressible payload.

B. Execution Time

Measuring the time required to store or encapsulate data is crucial for understanding performance of the system especially in applications consisting of endpoints with limited resources. Regarding this context, an experiment was conducted to compare the time required to store payload in blockchain, time required to store data in the ledger, and the time required to encapsulate payload using UXP technology. Fig. 15 shows the time required to store incompressible and compressible payloads in blockchain, encapsulated via UXP, and stored in the ledger respectively.

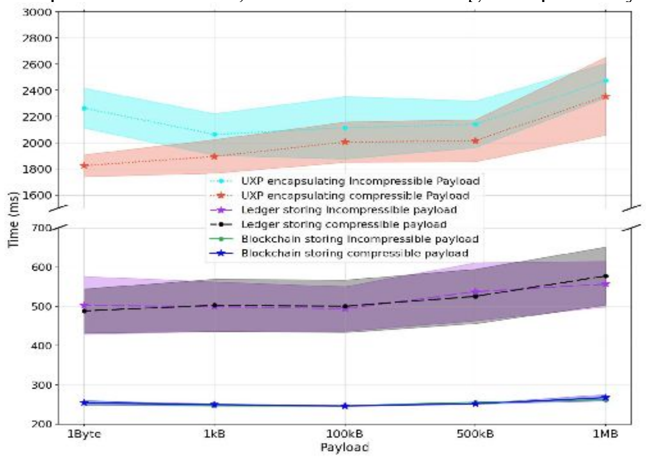


Figure 15. Blockchain vs UXP vs ledger in terms of time required to store payload.

As noted, the time to encapsulate data using UXP is more than 7 times greater than the time it takes to store data in the blockchain. Similarly, the time taken by UXP technology is more than 3 times greater than the time it takes to store data in the ledger. This disparity in execution time between the blockchain and ledger can be attributed to the computational requirements of PoW, which requires calculating a hash with specific difficulty level. This task poses a challenge for devices with limited resources and computing power.

Interestingly, the time difference between storing 1 Byte of payload and 1 MB of payload in the blockchain, on average, was just 7.15ms and 13.25ms. This indicates that the time required to store data in the blockchain is relatively independent of payload size.

Fig. 15 also highlights the narrowness of confidence intervals for time measurements of blockchain, indicating extremely low standard deviation and consistent time requirements regardless of payload size. Conversely, time measurements for UXP experiments have larger standard deviation, reflecting greater variability in the dataset. This could possibly stem from the different encryption and cloaking mechanisms used in UXP encapsulation. To depict this variability and randomness in the experiments, confidence intervals are included in the figures.

C. CPU Clock Cycles

CPU clock cycles directly influence the power consumption, heat dissipation, and resource allocation, in an embedded computing system, making it a crucial parameter to understand.

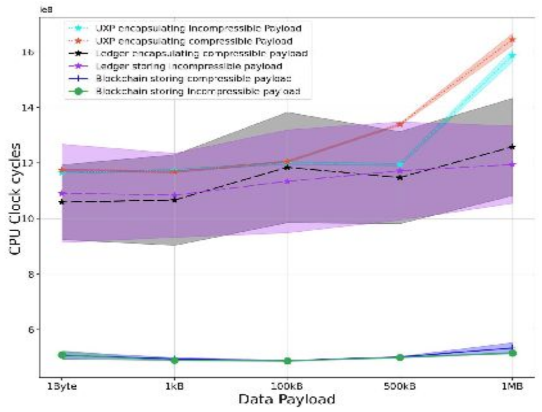


Figure 16. Comparison of CPU clock cycles for blockchain, ICTO, and ledger.

Fig. 16 compares CPU clock cycles for blockchain, UXP, and ledger technologies storing compressible and incompressible payload of various sizes.

As indicated in the Fig. 16, the CPU clock cycles required for UXP encapsulation were 2 times greater than storing data in blockchain. Interestingly, the CPU clock cycles required for UXP, and ledger were nearly equal for compressible and incompressible payloads.

D. Random Access Memory (RAM)

When choosing data protection technology, the balance of security, resource efficiency, and performance is important. The Resident Set Size (RSS), which measures the amount of physical RAM consumed by a process [25, 26] is a useful indicator of memory utilization for blockchain, ledger, and UXP creation on an individual computer.

Fig. 17 shows the average RSS values for compressible and incompressible payloads encapsulated via UXP or stored via blockchain or ledger. This data clearly illustrates that, for the same payloads, RSS required for blockchain, and ledger is 50% greater than RSS required for UXP encapsulation, regardless of payload type.

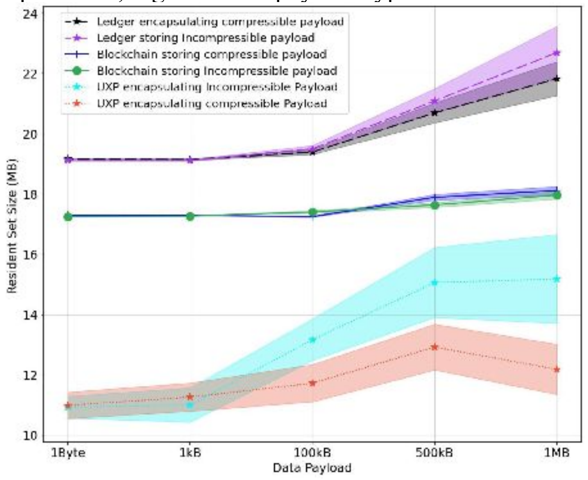


Figure 17. RSS memory comparison for blockchain, ledger, and ICTO.

E. Network Analysis

Network performance for blockchain and ICTO transferred via TCP was evaluated using identical client and server systems via data gathered on the client side. Blockchain payloads were compressed using the LZ77 algorithm and encrypted with ECC encryption prior to transmission. UXP objects were created using compressible payloads.

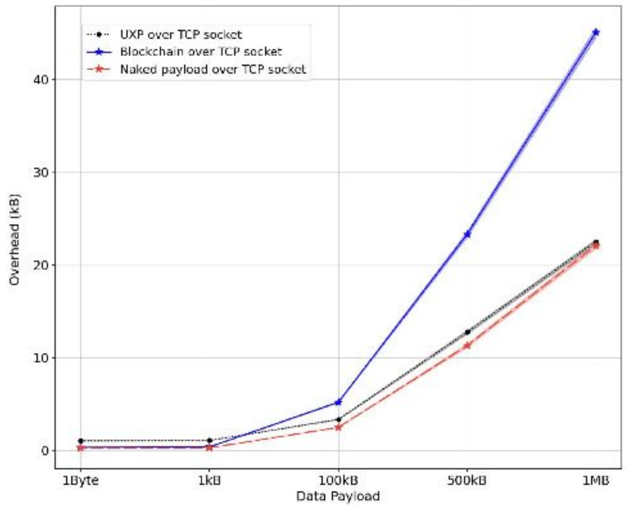


Figure 18. Cumulative Header Size vs Payload Size.

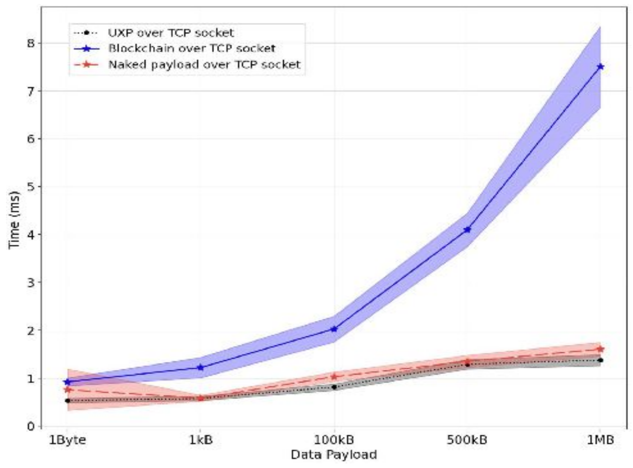


Figure 19. Network Latency vs Payload Size.

The overhead for plain TCP connections (denoted “naked payload” in the figures) and UXP/TCP was found to be similar for larger payloads as seen in Fig. 18. The overhead for blockchain was found to be significantly higher than for plain TCP and UXP/TCP for large payloads. This can be attributed to the relatively larger blockchain blocks vs. plain TCP and UXP/TCP. The encryption overhead of the blockchain blocks also increases with payload. However, the overhead of UXP/TCP is constant with mean value of 25.20 kB.

Fig. 19 illustrates that the total time required to transport UXP/TCP was surprisingly efficient as compared with blockchain, as block transport required as much as 500% of the time required for UXP/TCP.

F. Memory Footprint

The memory footprint of UXP encapsulation compared with encrypting blocks of blockchain with Ascon is also important for IoT applications. The Ascon encryption algorithm is designed for lightweight usage and easy implementation with minimal overhead [29]. “Ascon-128” with key size of 16 bytes was used in this research to encrypt individual blocks of the blockchain. The memory footprint of the resulting block was compared with UXP objects for the same payload.

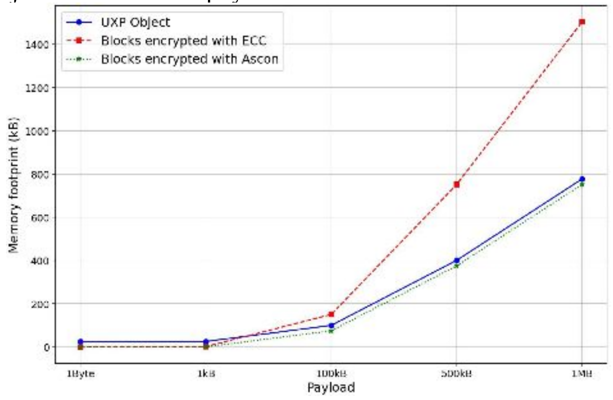


Figure 20. Memory footprint comparison for compressible payloads.

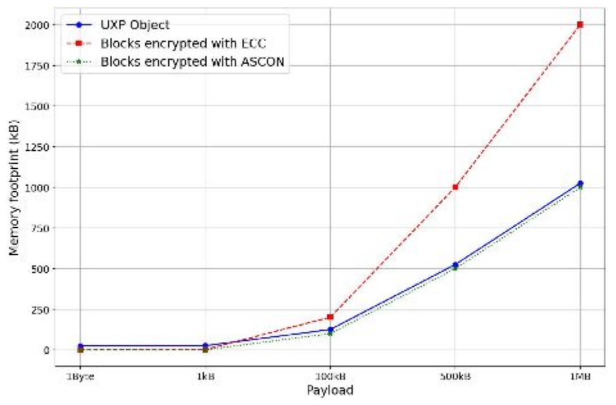


Figure 21. Memory footprint comparison for incompressible payloads.

Fig. 20 and Fig. 21 comparison memory requirements of UXP encapsulation and blocks of blockchain or ledger encrypted by ECC and Ascon algorithms for compressible and incompressible payloads respectively.

From the figures, it is clear that the memory requirements of UXP and Ascon encrypted blocks were almost identical for larger payloads. However, this requirement for blocks encrypted by ECC increased with payload size. Ascon is specifically designed for constrained implementation and low memory footprint. The fact that UXP memory requirements are similar to Ascon is surprising, particularly considering that UXP encapsulation includes multiple layers of encryption.

VI. Security Analysis

A major part of this research is concerned with the investigation of strengths and weaknesses of ICTO technology as a data security measure. Assessment of the security provided by the technology through conventional cryptanalytic techniques as well as modern approaches such as machine learning is a particularly important aspect of the investigation.

Cryptanalysis is the study and practice of analyzing data and cryptosystems for weaknesses and vulnerabilities that may be used to extract useful information [30, 31]. Two main categories of cryptanalysis are symmetric cryptanalysis for symmetric ciphers, and asymmetric/public key cryptanalysis for asymmetric ciphers. Some common symmetric cryptanalytic techniques include brute force attacks, differential cryptanalysis, and algebraic attacks [31-34]. Common techniques for public-key cryptanalysis include factoring attacks and discrete logarithm problem solving [35-37]. Cryptanalysis also relies on the availability of related information, such as the cryptographic algorithm applied, the plaintext used to generate ciphertext, and the ciphertext itself.

Unlike cryptanalysis of symmetric or asymmetric encryption algorithms, where a string of plaintext of a given length results a ciphertext of comparable length, ICTO objects typically have a minimum size of 25 kilobytes regardless of payload size. This makes it difficult to determine where the payload is located within the object. UXP objects were therefore analyzed with the assumption that the plaintext is not available. This approach mimics the role of an attacker who can observe UXP objects in flight or at rest.

As a preliminary measure, a large number of UXP objects created using the same XML policy file and user data were hashed and the resulting hashes were compared with the aim of finding a collision/repetition. None of the resulting hashes matched, which suggests that every UXP object is unique regardless of embedded user data or policy.

A. *Frequency Analysis*

In classical cryptanalysis, frequency analysis (letter counting) is the study of frequency of occurrences of letters or group of letters in ciphertexts [30-32]. It is one of the most basic and common methods to analyze ciphertexts and has been used for breaking many classical ciphers.

Frequency analysis was performed for a large number of UXP objects (n=500) protecting user payload. Characters were read at each index/offset/position in the UXP file, and the number of occurrences were tabulated. This analysis was performed individually for each object file to obtain a scatter plot, and then repeated for the entire set of files to obtain an average character-frequency plot.

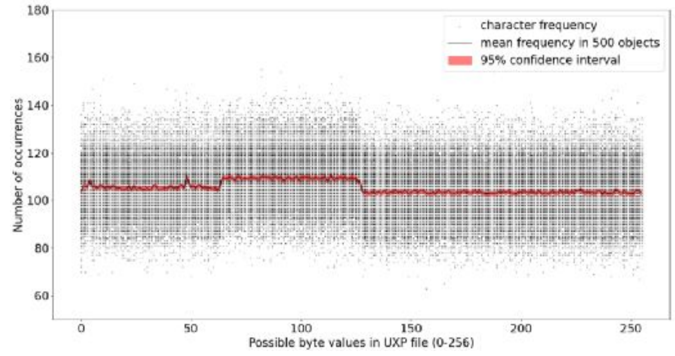


Figure 22. Plots showing byte value of characters vs. frequency of occurrence. The points plotted in black represent character frequency for individual files. The line surrounded by red band represents 95% C.I. plot of average frequency of characters in all 500 object files.

Fig. 22 shows the result of frequency analysis through a scatter plot and a line plot. Each black mark in the scatter plot represents the frequency of corresponding character (expressed in base 10) for a single UXP object. Each character is represented with an 8-bits byte and thus the character pool has 256 possible values. The composite scatter plot contains data for 500 UXP objects.

The red line in Fig. 22 displays the 95% confidence interval for each character value and location. Notably, the scatterplot indicates that regardless of character position, the distribution of values is distinctly non-uniform. This is an unexpected and potentially problematic outcome.

Characters with base-10 values greater than 63 and less than 127 are observed to have higher frequency of occurrence. Also, slightly higher frequency of occurrence near 105 is observed for characters 0 to 63 compared to characters 127 and above which have an average frequency of about 103. This observation counters the intuitive notion that character values in the UXP objects would be uniformly distributed regardless of position.

B. *Positional Analysis*

As a measure of finding a structure or similarity that may be common in UXP files, coincidence counting – a technique of putting two or more texts side-by-side and recording the number of times and position where identical characters repeat was performed for several UXP objects. It was observed that sets of characters at positions 492 to 494 (3 characters) and at positions 503 to 512 (10 characters) repeat for any UXP object. This observation along with the evidence of non-uniformly distributed character sets led to the analysis of characters based on their position/index in UXP objects.

A scatter plot of character position vs. value using 150 UXP object files all protecting 1 byte of user data is presented in Fig. 23. For simplicity, only the first 700 positions of each UXP file are considered in the plot. It can be observed from the plot that the characters within a range of positions always have a value within a fixed range. This outcome is concerning and non-intuitive for a collection of encrypted/cloaked segments of data.

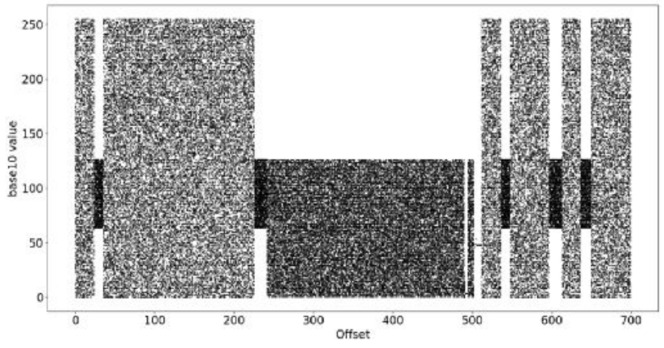


Figure 23. Scatter plot of character position vs. character value for 150 UXP files. The first 700 positions are considered for each object file.

Characters having base-10 values in range of 0-225, 64-127, and 1-126 are observed to be occurring in fixed ranges of positions. This pattern was observed throughout the entirety of UXP files. However, such patterns occur more frequently within the first 650 positions. Although it is not ideal for ciphertexts to contain a fixed set of characters occurring at a given range of positions, UXP objects are observed to have a clearly defined pattern/structure. These patterns were found to be distributed throughout the entirety of every UXP files regardless of variations in policy parameters set in the XML file.

Through careful analysis, it was found that some of the positions in the UXP files always contain a fixed set of characters. This is seen specifically for position 491 which contain only 14 possible characters. Positions 492 to 494 always contain a character with base-10 value of 4. Similarly, positions 503 to 512 always contain the same character. This data validates observations and inferences from frequency analysis because these characters evidently have a higher frequency of occurrence and are always present in several fixed positions of UXP objects. Such a composition of characters observed in UXP objects were found to remain unaffected even when the original XML file (containing policy specification) or payload was modified. This outcome is concerning and counter-intuitive for a collection of data which is encrypted or cloaked.

C. *Entropy Analysis*

Different levels of "surprise," "uncertainty," or "information" can be expressed in the information theory metric of entropy [31]. Entropy is the expected value or mean of the "information function" of the probability distribution for a set of characters. It indicates the "uncertainty" of a subsequent realization from a random source with a certain probability distribution. For UXP objects, the entropy for each position in the object yields an estimate of the number of bits required to store or transmit the information contained. Alternately, the positional entropy can measure the uncertainty related to a particular character in each position in the UXP object.

To calculate the positional entropy, character occurrences for each position in 500 UXP objects protecting 1 byte of user data were recorded. Kernel density estimation (KDE) [37-40] was employed to approximate the probability density of sample data recorded for each position and used to calculate the positional entropy for the first 700 bytes of each UXP object.

Fig. 24 presents entropy vs. unit index plot for the first 700 bytes of 500 UXP files. The X-axis contains 700 units and 350 units in the bottom and top axes to represent 1-byte units and 2-byte units respectively. Note that each 1-byte unit contains 8-bit characters (256 possible values), and each 2-byte unit contains 16-bit characters (65,536 possible values).

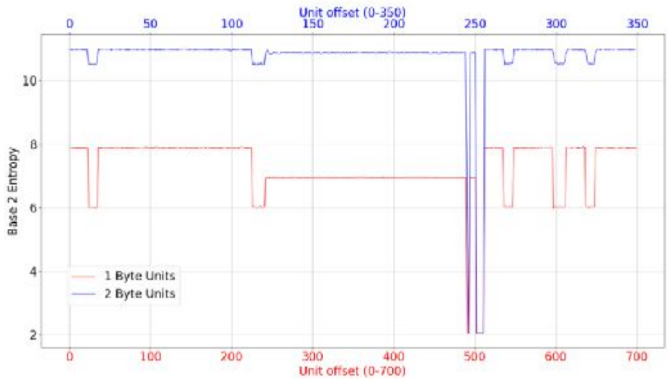


Figure 24. Line plot of entropy vs. unit index. Entropy of 1-byte units (red) and 2-byte units (blue) are shown. The vertical axis is presented using the base-2 entropy values for each unit offset.

The plots in Fig. 24 show large variations in entropies with respect to unit offsets/indices. Dips in entropy values are observed in indices where a fixed subset of characters were found to occur in Fig. 23. Note that the maximum value of base-2 entropy for an 8-bit index is 8 and for a 16-bit index is 16. This indicates the number of bits required for lossless transmission of information carried by a single unit. Most positional entropy values are maximized, but the regularity of "dips" in entropy are concerning.

Dips in entropy are observed in unit indexes where character groups appeared to “cluster” in Fig. 23. For instance, in the 1-bye unit vs. entropy plot, dips in entropy values are seen in the positions/offsets 25 to 36, 227 to 242, and so on. The trends observed in Fig. 24 correlated specifically with the constrained character occurrences observed in Fig. 23.

Lower entropy values indicate that the possible outcomes of a random variable or data source (characters occurring at given positions in this case) have a non-uniform probability distribution. In other words, the probability distribution may be skewed/warped heavily towards a single outcome. This result indicates that UXP objects are “leaking” information at well-defined locations.

In contrast, the positional entropy of ciphertexts generated from random data samples using AES and 32-bit keys is presented in Fig. 25.

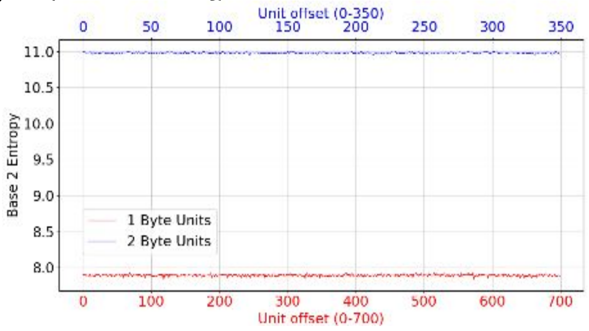


Figure 25. Line plot of entropy vs. unit index. 1-byte and 2-byte units were taken to obtain line plots in red and blue colors respectively.

Unlike the “dips” in positional entropy for UXP objects, AES-encrypted data produces consistent positional entropy of around 7.9 and 11 for 1 byte and 2-byte units respectively, regardless of character position. Thus, characters at each index of the ciphertexts are uniformly distributed and hence do not exhibit any kind of structure, or potential data leakage.

D. Cryptanalysis Using Unsupervised Machine Learning

To analyze patterns in UXP objects using different unsupervised ML techniques, a dataset containing relevant features was created using a large number of objects. The base-10 character values in each position from each object was recorded, and features of each array were extracted. Five characteristics including largest value, smallest value, mean, difference of largest and smallest values, and entropy were recorded or calculated. For simplicity, only the first 700 positions of the objects were considered. As a result, 700 records each with 5 features are present in the final dataset to be submitted to ML algorithms. The accuracy of the features depends upon the number of objects used for calculating each feature value.

Two popular unsupervised ML algorithms, k-means and agglomerative clustering were used to discover clusters of similar orientations in the set of UXP objects.

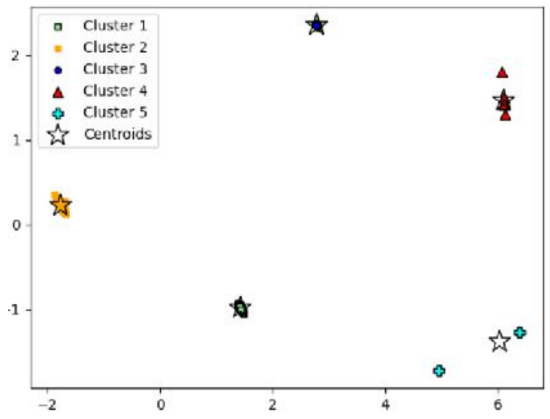


Figure 26. Clusters visible with k-means clustering.

Fig. 26 indicates that 5 distinct clusters with high degree of separation are present in UXP objects. These clusters indicate that distinct patterns or structures are present in every UXP object. The frequency of such patterns/structures in the objects correspond to the number of data points in each cluster. Information of this nature could be relevant for attackers because patterns in data can potentially expose or leak information and may act as the weakest points of attack.

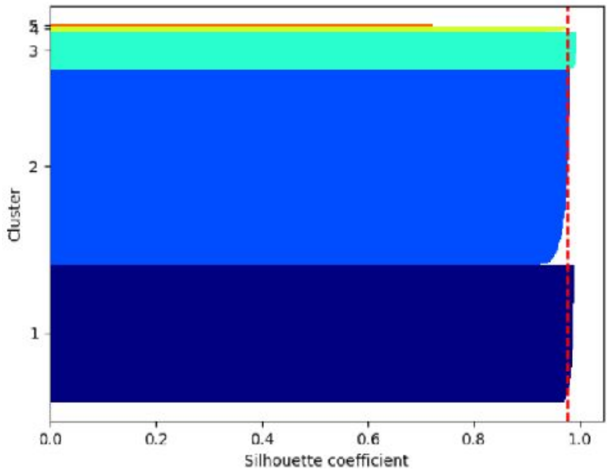


Figure 27. Silhouette plot of resulting k-means clusters.

Fig. 27 shows the silhouette plot obtained using data points from each cluster, which graphically depicts how well data points fit into the clusters to which they have been assigned, as well as the quality of separation. Silhouette values signify good or bad clustering with a range of [-1, 1]. The mean silhouette value in Fig. 27 is roughly 0.9 which is near the maximum value of 1.

Combining observations from Fig. 26 and Fig. 27, it is clear that UXP object have at least 4 distinct patterns of data within them. These inferences also align well with prior observations. As a result, it seems clear that more sophisticated or in-depth machine learning processes may reveal additional information about the UXP objects, or the data payloads contained within them.

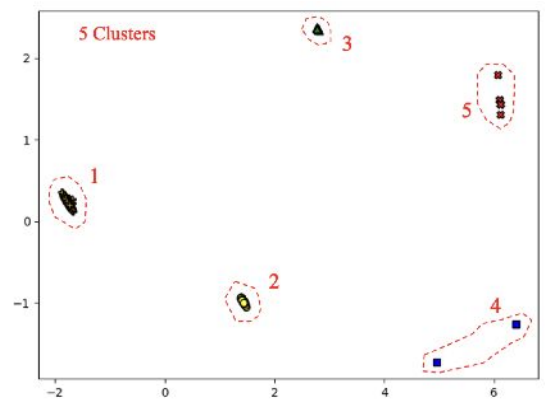


Figure 28. Clusters obtained using agglomerative clustering.

The presence of distinct clusters or groups of data inside UXP objects is further supported by Fig. 28, obtained by using agglomerative clustering. Similar to the results shown in Fig. 26, at least 5 different clusters are observed using this ML method.

Both approaches suggest that UXP objects consist of distinct patterns or structures which are evident when observed with respect to the position or index of characters. This characteristic of UXP objects exposes potential vulnerabilities that may be determined or exploited using more sophisticated ML approaches.

VII. Conclusion

This research explores the potential of the ICTO concept as a data security technology for IoT and general-purpose computing. Comparative performance with popular solutions or technologies is presented, including evaluations with TLS for network transmission and blockchain for storage and security. Based on the outcomes of this study, the ICTO concept could be an alternative to conventional security techniques, especially for IoT. Protecting data at the point of origin or the source itself with access and authorization policies embedded to the data itself seems to be the prime advantage of the ICTO concept. Most security frameworks rely on separate mechanisms to protect information when it is at-rest or in-flight, and these mechanisms have to be tightly coupled to provide comprehensive, end-to-end security. Such integrations are not only complex and costly but also introduce vulnerabilities that can exploited. The ICTO concept is a bold approach that may address many issues associated with data security.

A. *Performance*

System performance results in Section IV show that it takes at least 1 second of user time and around 47 MB of memory for UXP object creation. Transport of UXP objects over TCP is observed to require higher CPU and memory resources than plain TCP for small payloads. However, for larger payloads, transport of UXP objects was found to require fewer system resources compared to transport of raw user data over plain TCP. System resources used for transporting data via TLS over TCP were significantly greater than UXP object transport, requiring 50% more CPU cycles and 20 MB more memory. Automatic lossless compression of text payloads resulted in objects with total size smaller than the original payload.

The transport of user data via TLS/TCP required significantly greater transmit time and control plane overhead than the other two approaches, regardless of the payload size. In comparison, UXP/TCP required slightly more time and overhead for small payloads but outperformed other methods for large payloads.

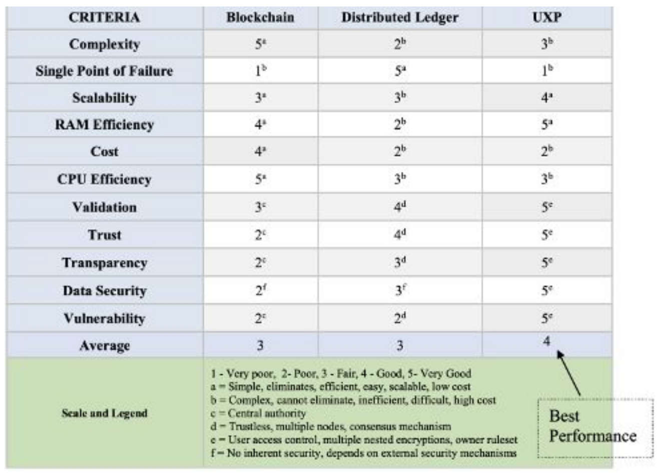
B. *Blockchain vs ICTO*

Scalability is a key consideration in data security applications, and technologies such as blockchain have clear scalability issues because of the cumulative nature of the chain. In contrast, ICTO (as realized as UXP) creates an independent object for a given payload with overlapping layers of security and relatively constant overhead.

In terms of memory and storage utilization and clock cycle requirements, ICTO provides a substantial, relatively deterministic outcome. This result is in stark contrast with an implementation of blockchain in a modified distributed ledger with proof-of-work. This suggests that ICTO has the potential to replace the complex distributed ledger technologies employed in various applications.

Table 1 summarizes critical system performance criteria including literature survey discussion and experimental results using a modified Likert Scale [40] or Mean Opinion Score (MOS) [41] to quantify and rank certain qualitative results.

TABLE 1. Comparison of blockchain, ledger, and UXP



CRITERIA	Blockchain	Distributed Ledger	UXP
Complexity	5^a	2^b	3^b
Single Point of Failure	1^b	5^a	1^b
Scalability	3^a	3^b	4^a
RAM Efficiency	4^a	2^b	5^a
Cost	4^a	2^b	2^b
CPU Efficiency	5^a	3^b	3^b
Validation	3^c	4^d	5^e
Trust	2^c	4^d	5^e
Transparency	2^c	3^d	5^e
Data Security	2^f	3^f	5^e
Vulnerability	2^c	2^d	5^e
Average	3	3	4
Scale and Legend	1 - Very poor, 2- Poor, 3 - Fair, 4 - Good, 5- Very Good a = Simple, eliminates, efficient, easy, scalable, low cost b = Complex, cannot eliminate, inefficient, difficult, high cost c = Central authority d = Trustless, multiple nodes, consensus mechanism e = User access control, multiple nested encryptions, owner ruleset f = No inherent security, depends on external security mechanisms		Best Performance

C. *Security of ICTO*

Basic cryptanalysis techniques including frequency analysis, index-of-coincidence and positional entropy revealed potential patterns or structures within UXP objects. Frequency analysis of UXP objects indicates that some characters occur more frequently than others and are not uniformly distributed for every position in UXP objects.

Some of the positions were found to always contain characters from a fixed subset of possible characters, and positional entropy analysis revealed that UXP objects follow a fixed structure or pattern to store information within them. The presence of consistent structures throughout UXP objects indicates that they are leaking information that may serve as a starting point for a more sophisticated attack.

Unsupervised ML approaches confirmed and reiterated the fact that patterns are present in the UXP objects. ML algorithms were successful in finding distinct clusters with high degree of separation. The clusters indicate that at least 4 different patterns or structures can be found in the first 700 bytes of every UXP object.

D. Summary

UXP as an implementation or realization of ICTO technology is a complete package that focuses on data security with enhanced protection schemes. The UXP instantiation of ICTO is a proprietary implementation. As such, specific implementation details are not available for evaluation. However, as an instance of a new class of data security techniques, the contrasts and comparisons with similar and enabling technologies is valuable. Additional research through cryptanalysis is necessary to comprehend the extent of data security actually provided by the UXP implementation of ICTO. But what's clear is that it is a ready-to-use approach to data security which intrinsically supports lossless compression for implementation efficiency. It is extremely useful in cases where security and scalability are vital. Based on the results and inferences drawn from this work, the ICTO concept indeed has a potential to be a useful technology for securing data in IoT as well as general purpose computing.

Even though the UXP implementation of ICTO which was leveraged in this research may not be fully optimized, the performance statistics are compelling, especially when evaluated in context with conventional in-transit data protection schemes. For use cases in IoT, while the findings indicate the overhead of UXP or ICTO is not optimal, the constant overhead even for larger payloads does show potential. The promising trajectory of ICTO with its unique data security approach, and scalability model, justifies further exploration in IoT domain. Further research to optimize ICTO could prove beneficial especially for IoT where securing data right from the source is often difficult.

VIII. Future Work

Clearly, a wider range of ML techniques and exploration of deep learning methodologies could prove to be fruitful in the analysis of ICTO. This work has shown that basic ML methods can detect multiple structures within the secure objects, but the importance of these structures in data leakage is unclear.

Experimental analysis of the ICTO implementation suggests that it is also worth considering an open-source implementation, which could provide notable advantages including transparency, auditability, and interoperability in applications where data security is a critical requirement.

IX. References

[1] B. Thapa, B. Sharma, and S. McClellan, "Comparative Performance of TCP and MQTT," in Proc. 18th Int'l Conf. on Digital Telecom (ICDT 2023), pp.10-14, Venice, Italy, Apr. 2023. ISBN: 978-1-68558-034-6

[2] H. Dai, J. Xu, and Q. Li, "Enterprise Cloud Computing Adoption: An Empirical Study of Factors Influencing Adoption," Info. Sys. Frontiers, vol. 17, no. 2, pp. 243-257, 2015.

[3] T. Erl, Z. Mahmood, and R. Puttini, "Cloud Computing: Concepts, Technology & Architecture," Prentice Hall, 2013.

[4] M. Armbrust, A. Fox, et al., "A View of Cloud Computing," Comm. Of the ACM, v.53, n.4, pp. 50-58, 2010.

[5] C. Wang, K. Ren, W. Lou, and J. Li, "Toward publicly auditable secure cloud data storage services," in IEEE Network, vol. 24, no. 4, pp. 19–24, 2010.

[6] L. Wei, H. Zhu, et.al., "Security and privacy for storage and computation in cloud computing," Info. Sciences, vol. 258, pp. 371–386, 2014.

[7] S. Aldossary and W. Allen. "Data security, privacy, availability and integrity in cloud computing: Issues and current solutions." Int'l J. Adv. Comp. Sci. Appl. 7.4, 2016.

[8] Cloud computing-The business perspective. Decision support systems, vol. 51, pp. 176 – 189, 2011.

[9] S. V. Kartalopoulos, "Differentiating Data Security and Network Security," in IEEE Int'l Conf. Comm., Beijing, China, pp. 1469-1473, 2008.

[10] S. Turner, "Transport layer security," IEEE Internet Comp. vol. 18, n.6, pp. 60-63, 2014.

[11] K. H. Mohammed, A. Hassan, and D. Y. Mohammed, "Identity and Access Management System: a Web-Based Approach for an Enterprise," 2018.

[12] E. Biham and O. Dunkelman, "Cryptanalysis of the A5/1 GSM stream cipher," in Prog. Cryptology—INDOCRYPT 2000: 1st Int'l Conf. Crypt. In India, Calcutta, India, December 10–13, 2000. V.1. Springer, 2000.

[13] N. Novaes Neto, S. Madnick, M. G. de Paula, and N. M. Borges, "A case study of the Capital One data breach," 2020.

[14] W. Connelley and B. Gudaitis, "Architecture containing embedded compression and encryption algorithms within a data file," U.S. Patent 20030204718A1, Oct. 10, 2003.

[15] R. Patawaran and G. Chapman, "Secure storage and retrieval of confidential information," U.S. Patent 20110252480A1, Jul. 16, 2013.

[16] "What is Blockchain Technology", IBM Blockchain https://www.ibm.com/topics/what-is-blockchain

[17] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System". 2009. Bitcoin.org

[18] H.-N. Dai, Z. Zheng, and Y. Zhang, "Blockchain for Internet of Things: A Survey," IEEE Internet Things J., vol. 6, no. 5, pp. 8076–8094, Oct. 2019, doi: 10.1109/JIOT.2019.2920987.

[19] M. Samaniego, U. Jamsrandorj, and R. Deters, "Blockchain as a Service for IoT," in 2016 IEEE Int'l Conf. Internet of Things (iThings), Green Comp. Comm.(GreenCom), Cyber, Phys. Social Comp. (CPSCom), and Smart Data (SmartData), Dec. 2016, pp. 433–436. doi: 10.1109/iThings-GreenCom-CPSCom-SmartData.2016.102.

[20] P. Arul and S. Renuka, "Blockchain technology using consensus mechanism for IoT- based e-healthcare system," IOP Conf. Series: Materials Sci. and Eng., Feb. 2021. doi: 10.1088/1757-899X/1055/1/012106.

[21] S. Kim, G. Chandra Deka, and P. Zhang, "Role of Blockchain Technology in IoT Applications" in Advances in Computers, v. 115, Academic Press, Sep. 2019. [Online]. Available:

https://learning.oreilly.com/library/view/role-of-blockchain/9780128171929/S0065245819300397.xhtml

[22] R. Li, T. Song, B. Mei, H. Li, X. Cheng, and L. Sun, "Blockchain for Large-Scale Internet of Things Data Storage and Protection," IEEE Trans. Serv. Comput., vol. 12, no. 5, pp. 762–771, Sep. 2019, doi: 10.1109/TSC.2018.2853167.

[23] G. Smith, M. Weed, D. Fischer, and E. Ridenour, "System and methods for using cipher objects to protect data," U.S. Patent 11,093,623 B2, 2021.

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

[25] Canonical, "Forkstat - a tool to show process activity", https://manpages.ubuntu.com/manpages/xenial/en/man8/forkstat.8.html (accessed May 29, 2023).

[26] "perf stat," Linux manual page, https://man7.org/linux/man-pages/man1/perf-stat.1.html (accessed May 29, 2023).

[27] alexmgr, "tinyec." May 13, 2023. Accessed: May 20, 2023. [Online]. Available: https://github.com/alexmgr/tinyec

[28] "scipy.stats.t — SciPy v1.10.1 Manual." https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.t.html (accessed Jun. 10, 2023).

[29] "ASCON." Accessed: Apr. 25, 2023. [Online]. Available: https://ascon.iaik.tugraz.at/index.html

[30] B. Carter and T. Magoc, "Classical ciphers and cryptanalysis," in Space 1000, vol. 1, pp. 1, 2007.

[31] J. Dooley, "History of Cryptography and Cryptanalysis: Codes, Ciphers, and Their Algorithms," in History of Computing, Springer 2018.

[32] L. Knudsen and M. Robshaw, "Brute force attacks," in The Block Cipher Companion, 1st ed., New York, NY, USA: Springer, 2011, pp. 95-108.

[33] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," J. Crypt., vol. 4, no.1, pp. 3-72, 1991.

[34] N. Courtois and W. Meier, "Algebraic attacks on stream ciphers with linear feedback," in Proc. Int'l Conf. Theory and Appl. Crypto. Tech. - EUROCRYPT 2003, Warsaw, Poland, May 4-8, 2003, v.22, pp. 345-359, Springer, 2003.

[35] D. J. Bernstein et al., "Factoring RSA keys from certified smart cards: Coppersmith in the wild," in Proc. Int. Conf. Theory and Appl. of Crypt. and Info. Sec. - ASIACRYPT 2013, Bengaluru, India, December 1-5, 2013, vol.19, pp. 341-360, Springer 2013.

[36] K. S. McCurley, "The discrete logarithm problem," in Proc. Symp. Applied Math, vol. 42, pp. 49-74, 1990.

[37] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning," MIT press, pp. 73, 2016.

[38] M. Rosenblatt, "Remarks on some nonparametric estimates of a density function," in Annals of Math. Stat., vol. 27, no. 3, pp. 832-837, 1956.

[39] E. Parzen, "On estimation of a probability density function and mode," in Annals of Math. Stat., vol. 33, no. 3, pp. 1065-1076, 1962.

[40] R. Likert, "A Technique for the Measurement of Attitudes. Archives of Psychology," 140, 1–55, 1932.

[41] R. Streijl, S. Winkler and D. Hands. "Mean opinion score (MOS) revisited: methods and applications, limitations and alternatives." Multimedia Sys., vol. 22, pp. 213–227, 2016. https://doi.org/10.1007/s00530-014-0446-1