International Journal of Computer Applications (0975 - 8887)
Volume 187 - No.71, January 2026

Location-Dependent Cryptosystem: Geographically Bounded Decryption via UWB Timing-Encoded Key Reconstruction

Kunal Mukherjee
University of Evansville
1800 Lincoln Ave
Evansville, IN, USA

ABSTRACT

Digital content distribution and propitiatory research driven industries face persistent risks from intellectual property theft and unauthorized redistribution. Conventional encryption schemes such as AES, TDES, ECC, and ElGamal provide strong cryptographic guarantees, but they remain fundamentally agnostic to *where* decryption takes place. In practice, this means that once a decryption key is leaked or intercepted, any adversary can misuse the key to decrypt the protected content from any location.

This paper presents a location-dependent cryptosystem in which the decryption key is not transmitted as human- or machine-readable data, but implicitly encoded in precise time-of-flight differences of ultra-wideband (UWB) data transmission packets. The system leverages Ciholas DWETH101 hardware and a custom TiCK (Timing-encoded Cryptographic Keying) protocol to map a 32-byte SHA-256–derived AES key onto scheduled transmission timestamps. Only receivers located within a predefined spatial region can observe the packet timings that align with the intended "time slot" pattern, enabling them to reconstruct the key and decrypt the secret. Receivers outside the authorized region observe yield incorrect keys.

A complete prototype is implemented that encrypts and transmits audio data using our cryptosystem, and only when the receiver is within the authorized data they are able to decrypt the data. Our evaluation demonstrates that the system (i) removes the need to share decryption passwords electronically or physically, (ii) ensures the decryption key cannot be recovered by the eavesdropper, and (iii) provides a non-trivial spatial tolerance for legitimate users.

General Terms

Cryptography, Location-based Security, Wireless Systems, Secure Content Distribution

Keywords

Location-dependent cryptosystems, Ultra-WideBand (UWB), Geographically Bounded Decryption, Key Distribution, Secure Content Delivery

1. INTRODUCTION

Cryptographic primitives such as AES, TDES, ECC, and ElGamal have made it possible to protect data from digital piracy and intellectual property theft [7, 23]. These schemes are widely deployed across entertainment, cloud, financial, and research-driven industries [1, 2, 5, 6, 8, 10, 12, 14–19, 24]. However, while modern ciphers offer strong guarantees on confidentiality and integrity, they are fundamentally indifferent to the *geographic location* in which decryption occurs. Once an adversary obtains a valid decryption key, by intercepting it during transit or stealing it from an endpoint, there is nothing in the cryptosystem itself that prevents them from decrypting the protected content anywhere and at any time. These industries need new security measures to protect their intellectual property from corporate or international espionage.

This limitation is particularly problematic for domains where distribution is unavoidable but control is critical. Intellectual property theft occurs during data transfer [13]. For example, the entertainment industry routinely ships high-value content (e.g., pre-release films) to theaters or partners under contractual agreements, encrypting the media in transit and then sharing keys later. The standard pattern is to encrypt content with an industry-standard cipher such as AES-256 and deliver the decryption key via a separate channel, either electronically or physically [3]. This key-distribution step introduces a structural weakness: if the key is intercepted, copied, or misdirected, any unauthorized party can decrypt the content from any location.

Therefore, a viable solution is that the user should be able to decrypt a file only if he or she is at an acceptable geographic location pre-authorized by the sender. Therefore, this system will transfer the encrypted key so that only the user at the authorized location can decrypt it, without having prior knowledge of the sender or the password. Therefore, the need for sharing the password electronically or physically is omitted.

This intuition motivated the defense presented in this paper: designing a cryptosystem that ties decryption capability to a *geographic region*. Intuitively, we want a mechanism such that: (1.) a legitimate receiver located within an authorized region can reconstruct the decryption key automatically, without ever seeing the key in cleartext or manually entering a password, and (2.) any party outside that region, even if they capture all encrypted traffic, cannot recover the key from the signals they observe.

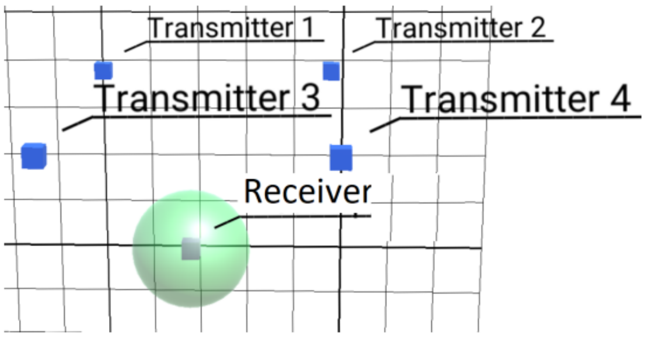


Fig. 1: Authorized-decryption geometry. Multiple UWB transmitters (anchors) define an authorized spatial region in which a receiver observes the intended time-of-flight–aligned packet timing pattern and can reconstruct the key for decryption.

Rather than transmitting a key as a conventional data payload or shared separately, we exploit precise timing information in ultra-wideband (UWB) radio signals. By mapping each byte of a 32-byte SHA-256–derived key to carefully scheduled packet transmission times, we ensure that only receivers experiencing the correct time-of-flight delays (i.e., those physically located in the intended region) reconstruct the intended key.
We design and implement a location-dependent cryptosystem built on Ciholas DWETH101 [4] hardware and a custom TiCK (Timing-encoded Cryptographic Keying) protocol. The sender encrypts data with AES-256 [20] using a password provided by the sender, derives a fixed-length key (32 bytes) by hashing the password using SHA-256 [21], and encodes that key across a sequence of 33 UWB transmissions. The mapping from password encrypted bytes to transmission timestamps is computed using:

(1) a global network time maintained by the server,
(2) a configurable initial offset that aligns with the server's internal timing windows,
(3) a minimum inter-packet delay to ensure receiver can process packets in real time, and
(4) per-transmitter time-of-flight offsets corresponding to the authorized decryption region.

A receiver within the authorized region (as shown on 1 observes packet arrival times whose time-of-flight align with the pre-defined "time slots" for each key byte, allowing it to reconstruct the SHA-256 key and decrypt the AES-encrypted payload. An eavesdropper at a different location, however, will observe shifted time-of-flight patterns; without precise knowledge of both reference timestamps and per-transmitter distances, they derive an incorrect key and fail to decrypt the content.
We implement a full prototype, where a UWB-based system encrypts audio files, transmits the key implicitly via packet timings, and automatically decrypts and plays the audio only within the authorized area. The prototype demonstrates that:

(1) the decryption password never appears in a human- or machine-readable form on the wire,
(2) the authorized region is a configurable *zone* rather than a single point, giving legitimate users spatial tolerance, and
(3) unauthorized receivers outside this zone do not reconstruct the correct key, even when they capture all packets.

To the best of our knowledge, we are the first work to explore how physical location can be elevated from an external policy constraint to a first-class factor in cryptographic key distribution. Instead of treating decryption keys as static secrets that must be guarded and transported, we embed the key into the physical properties of ultra-wideband (UWB) communication—specifically, the time-of-flight patterns observed only within an authorized region. The rest of this paper first provides the necessary background, then describes the system design and implementation in detail.

2. BACKGROUND

The current encryption standards for industries are AES256 (128 bits or higher), TDES (double-length keys), ECC (160 bits or higher) and ElGamal (1024 bits or higher) [1,2,5,6,8,10,12]. These encryption standards are secure as the underlying base problem, the discrete log problem (DLP), is intractable and exponentially hard for large primes [11]. Intractable is defined as taking thousands of years to brute force through the function, even for the top five supercomputers of the world.
In industry, the encryption standard used to encrypt data is AES [20]. The encrypted data is then transmitted using any one of the cryptographic network protocols, such as Internet Protocol Security (IPsec) [9]. However, the key of the encryption is independent of the location and it needs to be transferred to the receiver physically or electronically. Therefore, in this cryptosystem, to make sure the encrypted data can only be decrypted at the authorized location, the key of the encrypted data is associated with the approved location. The system will start the decryption process automatically without the need for the receiver's assistance, thus protecting the password's integrity.

3. SYSTEM DESIGN

3.1 Hardware

The cryptosystem requires a device that can emit ultra-precise timing information. Therefore, Ciholas' DWETH101 [4] was used as transmitter and NetApp server to relay DWETH101 timing information accurately up to eight nanoseconds. DWETH101 contains a DecaWave [22] chip that can emit timestamps accurately up to ten picoseconds. NetApp server uses the timestamp from DecaWave chip and makes it more precise by using the most significant bits to get information in the nanoseconds range or resulting in ± 3 cm precision in space.

3.2 Software

3.2.1 Audio Extraction. The software [1] was developed in multiple part and as an example demonstration an audio file was chosen, with the goal that the audio file will be decrypted only at the authorized located originally determined by the sender. Therefore, the development of an audio extraction tool was important. The audio extraction script takes an audio file, `input.wav`, and converts it into principal frequency values. The values are then stored in a file, which can later be used to convert the principal frequency values back into audio. The script uses the `ffmpeg` utility to convert the `.wav` file to principal frequency values.

3.2.2 AES-256 and SHA-256 Implementation. AES-256 encryption was used, and the encryption process operates on a specific

[1]The code is provided in `https://github.com/kunmukh/Location_Dependent_Cryptosystem`

buffer (e.g., 32 bytes). AES-256 encrypts the buffer using an initialization vector (IV) and the provided key. AES encryption produces 32 bytes, or one block, of encrypted data from the unencrypted buffer. The encrypted block is stored in a text file that would be transmitted later. This encryption process continues until all data has been converted into encrypted blocks.

The decryption process takes the encrypted data and the password and recreates the unencrypted data. The cryptosystem converts the user-provided password to a fixed 32-byte value using SHA-256. An SHA-256 hashing function has two important properties. First, it protects the integrity of the user's password since no one has knowledge of the plain text key except the user. Secondly, it creates a fixed length key every time, thus broadening the collision domain of the encrypted passwords.

3.2.3 *TiCK (Timing-encoded Cryptographic Keying) Protocol.*

The TiCK (Timing-encoded Cryptographic Keying) protocol uses the 32 bytes of the SHA password and transmits it via the timestamp difference. To transmit 32 bytes of SHA, 33 packets are needed, since the first transmission serves as a reference for the first SHA calculation.

A timestamp is the time the NetApp server transmits when one of the transmitters (i.e. DWETH101) connected to it receives or transmits a packet. If a device receives a packet, the reception timestamp is T_{rx}. Additionally, if the device transmits a packet, the timestamp is called the transmission timestamp, T_{tx}. The server runs an internal clock that provides time in Network Time (NT) ticks. K_f is a conversion factor that converts seconds to NT ticks and is numerically equal to 975000×65536, or $6{,}389{,}760{,}000$. One NT tick is equal to 16.5 nanoseconds or one second is 6,389,760,000 NT ticks. This shows how precise and accurate the NetApp server is.

The TiCK protocol had to be made compatible with the NetApp. If the protocol asked the NetApp to transmit one of its packets when the NetApp was scheduled to transmit one of NetApp's packets, then the TiCK packet would not be transmitted due to NetApp's internal packet taking higher priority. Thus, an initial offset had to be introduced for the protocol to work. This ensured the protocol did not interfere with the scheduling of the NetApp's internal packets and also marked the beginning of the 33-timestamp transmission sequence. The initial offset is called $T_{\text{startOffset}}$ and is numerically equal to 5 milliseconds or 319,488,000 NT ticks.

The *time window* is the amount of time after which the NetApp sends out its internal packets to make sure that all devices connected to it are time synchronized. This occurs after 100 milliseconds. Therefore, another offset, T_{net} or the network offset, is added so that the 33 transmissions can happen in one time window. The network offset is numerically equal to $n \times K_f$ NT ticks, where n represents a value that can be found by taking the current networking of the NetApp and dividing it by K_f. Hence, to begin a transmission and to give the reference for the first SHA calculation, a packet is transmitted at $T_{\text{net}} + T_{\text{startOffset}}$. $T_{\text{net}} + T_{\text{startOffset}}$ is called the *initial network cadence*. A transmission time window can be seen in Figure 2.

Once the protocol has determined which time window it is transmitting in as well as the first transmission packet, the protocol then schedules the next 32 packet transmissions so the differences can give the SHA back. The 32 packet transmissions must have at least a 2.5 millisecond gap between them. This is the time the devices need to process the received packet and return to listening mode. This time difference is called $T_{\text{betweenOffset}}$. $T_{\text{betweenOffset}}$ is known by both the receiver and the transmitter side. The transmitter also knows the time of flight a packet takes from each anchor to the approved location, T_{distA}.

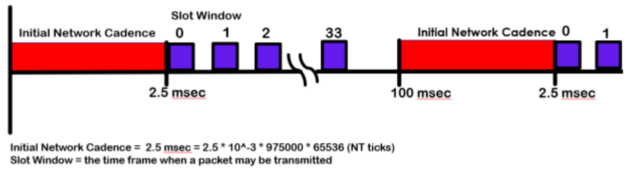

Slot Window
Initial Network Cadence
0
1
2
33
Initial Network Cadence
0
1
2.5 msec
100 msec
2.5 msec
Initial Network Cadence = 2.5 msec = 2.5 * 10^-3 * 975000 * 65536 (NT ticks)
Slot Window = the time frame when a packet may be transmitted

Fig. 2: NetApp timing window and network cadence. Illustration of the synchronization window in which the 33-packet TiCK transmission sequence is scheduled (starting at T_{net}+$T_{startOffset}$

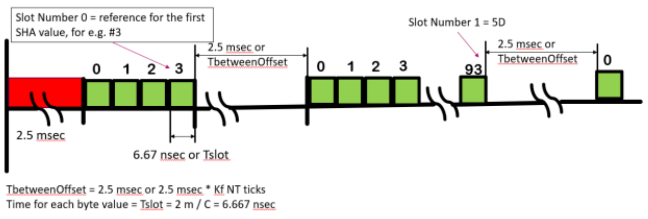

Slot Number 0 = reference for the first SHA value, for e.g. #3
0
1
2
3
2.5 msec or TbetweenOffset
0
1
2
3
Slot Number 1 = 5D
93
2.5 msec or TbetweenOffset
0
2.5 msec
6.67 nsec or Tslot
TbetweenOffset = 2.5 msec or 2.5 msec * Kf NT ticks
Time for each byte value = Tslot = 2 m / C = 6.667 nsec

Fig. 3: Slotting scheme for timing-encoded key bytes. A SHA-256 byte is mapped to a discrete slot number (0–255) and transmitted via an interpacket delay of $SlotNumber$ x T_{slot}, with T_{slot} chosen to match the desired spatial tolerance of the authorized region.

Next, the protocol considers the time allocated per slot corresponding to the SHA values, T_{slot}. T_{slot} depends on the amount of approved space of the decryption region. For example, a decryption region, a sphere of radius 2 meters, would be 2 meters divided by c or 6.667 nanoseconds. There are 256 time slots corresponding to the 255 values (0 to FF) that the SHA can take. A transmission slot window can be seen in Figure 3.

After determining all the constants, the TiCK protocol determines which values of the SHA need to be transferred. 5D will be used as an example. The transmission side will take the last transmission timestamp calculated, $T_{\text{tx}}(n-1)$, and add the time of flight of the packet from the last chosen anchor to the reception region, $T_{\text{distA}}(n-1)$. This gives the transmitter the reference point. Then, the transmission side adds $T_{\text{betweenOffset}}$. This ensures that the receiver is in the listening state. Then, the transmission side takes the SHA value `SlotNumber` 5D or 93 and multiplies it by T_{slot}. The final step is to subtract the time of flight, $T_{\text{distA}}(n)$, to account for the time it takes from this current packet to go from the current transmitter to the approved location. Therefore, the transmission equation is:

$$\begin{aligned} T_{\text{tx}}(n) = T_{\text{tx}}(n-1) + T_{\text{distA}}(n-1) + T_{\text{betweenOffset}} \\ + (\text{SlotNumber} \times T_{\text{slot}}) - T_{\text{distA}}(n). \end{aligned} \tag{1}$$

After the receiver accepts a second timestamp, the receiver first saves this timestamp as it becomes the reference for the second SHA transfer. Secondly, the receiver utilizes the second timestamp to get the first SHA value. The reception side takes the second timestamp and subtracts the first timestamp from it. Then, the reception side subtracts $T_{\text{betweenOffset}}$ from it. As a result, the receiver is now left with T_{slot} times the SHA value. Therefore, after dividing the result by T_{slot}, the slot value is equal to the first SHA transmitted value. Thus, the reception equation is the following:

$$\text{SHA value}(n) = \frac{T_{\text{rx}}(n) - T_{\text{rx}}(n-1) - T_{\text{betweenOffset}}}{T_{\text{slot}}}.$$

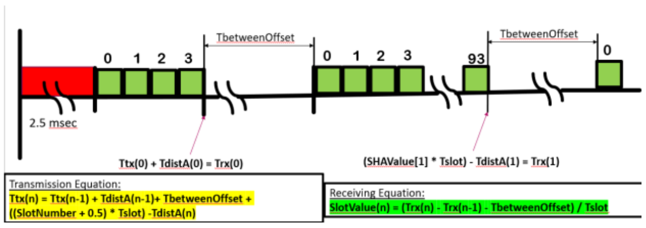

TbetweenOffset
0 1 2 3
0 1 2 3
93
TbetweenOffset
0
2.5 msec
Ttx(0) + TdistA(0) = Trx(0)
(SHAValue[1] * Tslot) - TdistA(1) = Trx(1)
Transmission Equation:
Ttx(n) = Ttx(n-1) + TdistA(n-1)+ TbetweenOffset + ((SlotNumber + 0.5) * Tslot) -TdistA(n)
Receiving Equation:
SlotValue(n) = (Trx(n) - Trx(n-1) - TbetweenOffset) / Tslot

Fig. 4: Transmit/receive timeline for decoding one key byte. Timeline view of the transmission and reception equations showing how the receiver cancels fixed offsets (e.g., $T_{betweenOffset}$) and uses the remaining delay to recover the slot value (and thus the SHA byte).

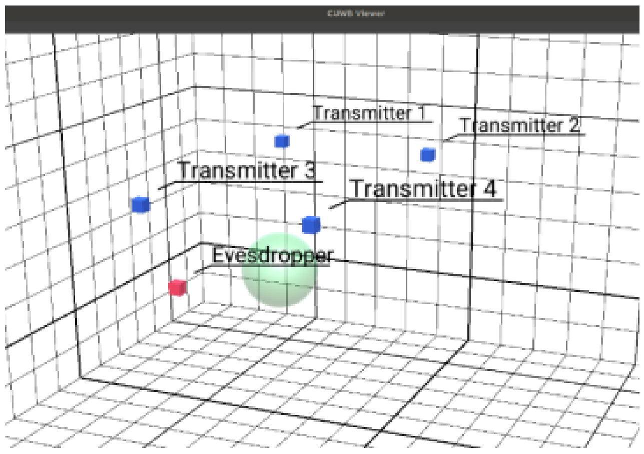


Fig. 5: Threat model: intended receiver vs. eavesdropper. An unauthorized receiver outside the approved region observes shifted time-of-flight timing, causing incorrect slot decoding and failure to reconstruct the correct key.

Figure 4 shows how the different constant parameters can be visualized in a reception timeline.

The eavesdropper is a receiver who is not at the approved location, represented in Figure 5. For the eavesdropper to obtain the SHA byte, they must overcome two security measures. First, the eavesdropper must know the reference timestamp, $T_{\text{rx}}(n-1)$, from which the second timestamp would be subtracted. Secondly, the eavesdropper must know the distances, $T_{\text{distA}}(n-1)$ and $T_{\text{distA}}(n)$, from which both packet transmissions are made. Without this information, the result of the second timestamp minus the first timestamp cannot be divided by the T_{slot} value to get the correct SHA value. From using Figure 6 and Figure 7, one can observe that being just 1.06 meters away from the approved location can change the SHA value.

3.2.4 Cryptosystem Simulator. The cryptosystem simulator was developed as a proof-of-concept to determine whether the transmission and reception equations will work. The simulator was built using TCP, which substituted UWB. Because the real-world system will have atmospheric noise, the simulation modified the reception equation by adding T_{noise} to account for it. As a result, the reception equation became:

$$\text{SHA value}(n) = \frac{T_{\text{rx}}(n) - T_{\text{rx}}(n-1) - T_{\text{betweenOffset}} + T_{\text{noise}}}{T_{\text{slot}}}.$$

The simulator server took an audio file, password, and the distances of the three anchors from the approved location. The server first generated an SHA hash of the password and then encrypted the audio file. Then, the server divided the encrypted data into 800-byte blocks and sent them as payload with each packet. The server also included the transmission timestamp as part of the payload. The transfer terminated once all encrypted data had been transferred. The first packet for the transfer, T_{net}, was set to $n \times K_f$ NT ticks.

The simulator client connected, and the transfer began. The client used the transmission timestamps to generate a 32-byte SHA-1 value. The client then used the SHA value to decrypt the received encrypted data. After the decryption process was over, the client played the decrypted data back. If the correct SHA had been transmitted, it would play the same audio back as intended by the server. If not, the client would just play noise, indicating that the decryption process was unsuccessful.

The cryptosystem simulator accounted for atmospheric noise and showed that, with UWB as the medium of transfer, the cryptosystem would function.

3.2.5 Cryptosystem Server and Client. The cryptosystem server listens to NetApp's timing packets and maintains a time window so it can transmit the 33 packets within that window. First, the crypto system takes the password from the sender and encrypts the data using the SHA of the password. Then selects an appropriate T_{net} as the first packet transmission time and transmits the first packet with an 800-byte payload of encrypted data. This process continues until all encrypted data has been transferred.

The approved receivers keep receiving the packets and use the timestamps of the received packets to create the SHA back. The receiver waits for 2 seconds. If the receiver receives no packets within 2 seconds, it stops listening and starts to generate a SHA to decrypt the received data. After the decryption process completes, the crypto system presents the data to the user (e.g., by playing the audio file back).

3.3 Social and Environmental Impact

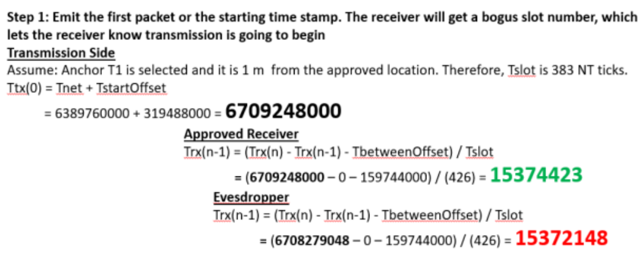
There are no social or environmental concerns associated with this project. This is a research endeavor to create a secure and robust cryptosystem. This will benefit everyone and improve the social aspect of sharing. This project will make sharing secure and trustworthy. Ciholas hardware is FCC approved, so the devices poses no environmental or health risks.

3.4 Industry Standards, Health and Safety Consideration

There are no comparable standards that map an encryption key to a location and transmit it wirelessly to the receiver. However, the user data will be encrypted before transmission using the industry-standard AES-256 encryption protocol. Therefore, AES-256 ensures that even if a packet is intercepted by an unintended receiver during transmission, the receiver cannot decipher it.

4. RESULTS

The cryptosystem demonstrated all requirements and features as stated during the development process. The cryptosystem was able to transfer the SHA of the password in a form that was neither machine-readable nor human-readable. The approved location was a space rather than a point. This ensured that the user had a certain degree of freedom and the location had a certain tolerance. The cryptosystem also started the decryption process without the



Step 1: Emit the first packet or the starting time stamp. The receiver will get a bogus slot number, which lets the receiver know transmission is going to begin

Transmission Side

Assume: Anchor T1 is selected and it is 1 m from the approved location. Therefore, Tslot is 383 NT ticks.

Ttx(0) = Tnet + TstartOffset

= 6389760000 + 319488000 = **6709248000**

Approved Receiver

Trx(n-1) = (Trx(n) - Trx(n-1) - TbetweenOffset) / Tslot

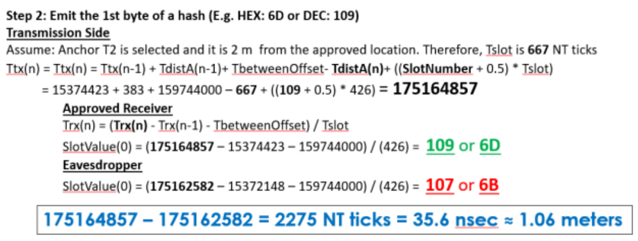
= (**6709248000** – 0 – 159744000) / (426) = **15374423**

Evesdropper

Trx(n-1) = (Trx(n) - Trx(n-1) - TbetweenOffset) / Tslot

= (**6708279048** – 0 – 159744000) / (426) = **15372148**

Fig. 6: Numerical example: reference mismatch outside the authorized region. Example calculation comparing the intended receiver's reference timing to an eavesdropper's; even small displacement changes the computed reference value used for slot decoding.



Step 2: Emit the 1st byte of a hash (E.g. HEX: 6D or DEC: 109)

Transmission Side

Assume: Anchor T2 is selected and it is 2 m from the approved location. Therefore, Tslot is **667** NT ticks

Ttx(n) = Ttx(n) = Ttx(n-1) + TdistA(n-1)+ TbetweenOffset- **TdistA(n)**+ ((**SlotNumber** + 0.5) * Tslot)

= 15374423 + 383 + 159744000 – **667** + ((**109** + 0.5) * 426) = **175164857**

Approved Receiver

Trx(n) = (**Trx(n)** - Trx(n-1) - TbetweenOffset) / Tslot

SlotValue(0) = (**175164857** – 15374423 – 159744000) / (426) = **109 or 6D**

Eavesdropper

SlotValue(0) = (**175162582** – 15372148 – 159744000) / (426) = **107 or 6B**

175164857 – 175162582 = 2275 NT ticks = 35.6 nsec ≈ 1.06 meters

Fig. 7: Numerical example: SHA-byte decoding divergence. Example showing that the intended receiver and eavesdropper decode different SHA values from observed timings, leading to different reconstructed AES keys and unsuccessful decryption outside the authorized region.

receiver's assistance, and any receiver not at the approved location did not receive the correct SHA value.

5. DISCUSSION

This study demonstrates that physical location can be elevated from an external policy constraint to a first-class input to key reconstruction by encoding a SHA-256, derived AES key into UWB packet timing patterns that are only decodable within an authorized region. While the prototype validates feasibility, several deployment and security questions naturally arise.

5.1 Design Trade-offs and Practical Constraints

A central design knob is the slot duration T_{slot}, which directly governs the spatial tolerance of the authorized region (e.g., a larger region implies a larger timing tolerance). In practice, this creates a location-utility trade-off: increasing tolerance reduces false rejections for legitimate receivers but also reduces the timing separation that protects against off-region reconstruction. The protocol also imposes real-time scheduling constraints (e.g., minimum inter-packet spacing so receivers can process packets), which affects throughput and robustness under load.

Does the security argument extend beyond a passive off-region eavesdropper?

The current rationale focuses on a passive adversary who records all packets but experiences different time-of-flight and therefore reconstructs an incorrect key. For stronger real-world assurance, the evaluation should explicitly consider (i) *replay* (retransmitting previously captured timing sequences), (ii) *relay/wormhole* (forwarding signals to emulate being in-region), (iii) *anchor compromise* (malicious transmitter emitting attacker-chosen timing), and (iv) *collusion* (multiple off-region receivers combining observations). Practical mitigations include time-bounded sessions (short validity windows tied to T_{net} cadence), per-session nonces folded into the key derivation, and transmitter authentication of the timing sequence (e.g., cryptographic tags over the intended schedule). These steps align with adversarial-evaluation practices commonly used in robust security research: explicitly enumerating attacker capabilities, then stress-testing the system under those capabilities rather than relying on a single threat model.

How should T_{slot} and the authorized region be selected in a principled way?

The work already links T_{slot} to region size through timing tolerance (e.g., meters divided by c), and uses 256 slots corresponding to byte values. A principled selection should incorporate measurement noise (clock jitter effects) and application requirements (acceptable false-reject rate). One robust approach is to treat slot selection as an optimization: choose T_{slot} to minimize off-region key success subject to an on-region decoding success constraint. Inspired by privacy-utility tuning in other security work (e.g., differential-privacy budget selection), T_{slot} can be viewed as a "physical tolerance budget" that must be justified empirically.

What are the scalability limits (latency/throughput) of a timing-encoded key channel?

The design transmits 32 SHA bytes using a 33-packet schedule (one reference plus 32 differences). This fixed overhead means key distribution cost is non-trivial if the system refreshes keys frequently or serves many receivers. Two ways to improve scalability: (i) transmit fewer bits per session by deriving multiple content keys from a single in-region "seed" (key hierarchy), and (ii) add redundancy selectively (error-correcting encoding across bytes) only when channel conditions degrade. These improvements mirror ideas from resilient distributed systems and robustness research: amortize expensive secure setup, and add redundancy only where the failure modes concentrate.

6. FUTURE WORK

Even with these security features, this system can be defeated by implementing many secondary anchors and using them to triangulate the approved location. The secondary anchors are used to determine which anchor sent which packets and their relative times of flight. Moreover, an important feature can be developed called a rolling key encryption system. With this, the receiver must immediately transmit a specific packet upon receiving an encrypted packet. The server will determine whether the timestamp for this specific packet is from the approved location. If not, the server will change the encryption key and re-transmit. If so, the next packet will be sent. This will continue until all of the packets have been transmitted and the cryptosystem can function normally.

7. CONCLUSION

By carefully scheduled UWB packet timing, this location-dependent cryptosystem enforces that decryption is possible *only* where the sender intends. Legitimate receivers within the authorized zone reconstruct the correct key and transparently decrypt audio content, while receivers outside this zone, even if they capture all packets, observe shifted timing patterns that yield incorrect keys. The system operates without ever exposing the password in human- or machine-readable form on the wire and initiates decryp-

tion without explicit user action, reducing both operational friction and key-handling risk.
More broadly, this cryptosystem suggests that cryptographic systems can benefit from tighter integration with the physical world: radio geometry, timing, and spatial constraints can all be treated as additional dimensions of access control. Future research may explore integrating location-dependent keying with hardware security modules, multi-factor attestation, or streaming telemetry, as well as formalizing security guarantees under stronger adversarial models. We hope this study provides both a concrete blueprint and a conceptual foundation for geographically bounded decryption as a practical complement to traditional cryptographic protections.

8. REFERENCES

[1] Ansi x9.52-2016: Triple data encryption algorithm modes of operation. Technical report, Accredited Standards Committee X9, Financial Industry Standards, 2016. Specifies Triple DES (TDES) modes widely used for data protection in the financial services industry.

[2] Payment card industry data security standard, version 4.0. Technical report, PCI Security Standards Council, 2022. Recommends strong cryptography such as AES and modern public-key schemes for protecting cardholder data across payment environments.

[3] Elaine Barker. Recommendation for key management part 1: General. NIST Special Publication 800-57 Part 1 Rev. 5, National Institute of Standards and Technology, 2020. Recommends that cryptographic keys (e.g., AES keys) be protected and distributed via secure key management mechanisms distinct from the channels carrying the encrypted data.

[4] Ciholas, Inc. Anchors. `https://cuwb.io/docs/v3.3/system-components/anchors/`, 2020. CUWB 3.3 (Bernoulli) Documentation.

[5] Morris Dworkin. Recommendation for block cipher modes of operation: Methods and techniques. NIST Special Publication 800-38A, National Institute of Standards and Technology, 2001. Defines standard modes of operation for block ciphers such as AES and 3DES for data protection in a wide range of applications.

[6] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985. Original description of the ElGamal cryptosystem, a building block for many practical public-key schemes used for confidentiality and authentication.

[7] Omar Flor-Unda, Lino Casado, Wilmer Aguilar, and María Muñoz. A comprehensive analysis of the worst cybersecurity incidents: Lessons learned. *Informatics*, 10(3):71, 2023.

[8] Ralph Holz, Johanna Amann, Olivier Mehani, Matthias Wachs, and Thomas C. Schmidt. A survey of SSL/TLS deployment on the internet. In *Proceedings of the 2011 IEEE Conference on Network and Service Management (CNSM)*, pages 163–170, 2011. Documents widespread deployment of block ciphers (e.g., AES) and public-key primitives (including ECC) in Internet-scale services such as web and cloud platforms.

[9] Stephen Kent and Karen Seo. Security architecture for the internet protocol. Request for Comments 4301, Internet Engineering Task Force (IETF), December 2005. Defines the IPsec architecture for providing confidentiality, integrity, and authentication for IP packets.

[10] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987. Introduces elliptic-curve cryptography, which is now widely deployed in protocols and products for secure communication and data protection.

[11] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. Reducing discrete logarithms in a finite field to discrete logarithms in a subgroup. *Journal of Algorithms*, 16(2):173–190, 1993. Analyzes the discrete logarithm problem and underpins the hardness assumptions used for discrete-log-based cryptosystems.

[12] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. Survey of cryptographic primitives such as block ciphers and public-key schemes and their deployment in real-world applications across multiple industries.

[13] Miller IP Law. The double-edged sword of technology transfer: Risks and rewards, 2024. Highlights that the risk of intellectual property theft increases when transferring technology, especially across borders.

[14] Kunal Mukherjee, Zachary Harrison, and Saeid Balaneshin. Z-rex: Human-interpretable gnn explanations for real estate recommendations. In *KDD Workshop on Machine Learning on Graphs in the Era of Generative AI (MLoG-GenAI)*, Toronto, Canada, 2025. Oral presentation.

[15] Kunal Mukherjee and Murat Kantarcioglu. Llm-driven provenance forensics for threat intelligence and detection. arXiv preprint / manuscript, 2025. Under submission; preprint available.

[16] Kunal Mukherjee, Joshua Wiedemeier, Qi Wang, Junpei Kamimura, John Junghwan Rhee, James Wei, Zhichun Li, Xiao Yu, Lu-An Tang, Jiaping Gui, and Kangkook Jee. Proviot: Detecting stealthy attacks in iot through federated edge-cloud security. In *Applied Cryptography and Network Security (ACNS)*, LNCS 14585, pages 241–268. Springer, 2024.

[17] Kunal Mukherjee, Joshua Wiedemeier, Tianhao Wang, Muhyun Kim, Feng Chen, Murat Kantarcioglu, and Kangkook Jee. Interpreting gnn-based ids detections using provenance graph structural features. 2023. Under submission; preprint available.

[18] Kunal Mukherjee, Joshua Wiedemeier, Tianhao Wang, James Wei, Feng Chen, Muhyun Kim, Murat Kantarcioglu, and Kangkook Jee. Evading provenance-based ml detectors with adversarial system actions. In *Proceedings of the 32nd USENIX Security Symposium*, Anaheim, CA, USA, 2023.

[19] Kunal Mukherjee, Jonathan Yu, Partha De, and Dinil Mon Divakaran. Provdp: Differential privacy for system provenance dataset. In *Applied Cryptography and Network Security (ACNS)*, 2025.

[20] National Institute of Standards and Technology. Advanced encryption standard (aes). Federal Information Processing Standards Publication FIPS 197, NIST, November 2001. Defines AES with key sizes of 128, 192, and 256 bits (AES-256).

[21] National Institute of Standards and Technology. Secure hash standard (shs). Federal Information Processing Standards Publication FIPS 180-4, NIST, 2015. Defines the SHA-1 and SHA-2 families of hash functions, including SHA-256.

[22] Qorvo, Inc. Dw1000: Ieee 802.15.4-2011 uwb wireless transceiver, 2025. Product page for the DW1000 ultra-wideband transceiver.

[23] Adriano Terra. Copyright law and digital piracy: An econometric global study. *North Carolina Journal of Law & Technology*, 17(4):563–626, 2016.

[24] Tianhao Wang, Simon Klancher, Kunal Mukherjee, Josh Wiedemeier, Feng Chen, Murat Kantarcioglu, and Kangkook Jee. Provcreator: Synthesizing complex heterogenous graphs with node and edge attributes. *arXiv preprint arXiv:2507.20967*, 2025.