Reducing the WSN's Communication Overhead by the SD-SPDZ Encryption Protocol

Alexander K. Alexandrov [1,*]

[1]Institute of Robotics, Bulgarian Academy of Sciences, Acad. G. Bonchev str., 1113 Sofia, Bulgaria

Abstract

Wireless Sensor Networks (WSN) have emerged as a pivotal technology in many application areas such as environmental monitoring, IoT, military applications, and healthcare. These networks consist of spatially distributed, autonomous sensors that cooperatively monitor physical or environmental conditions, such as temperature, sound, or pollution levels. The unique characteristics of WSNs, including their resource constraints (e.g., energy, memory, and computational capacity), make them vulnerable to various security threats. Information security in WSNs is crucial to ensure the confidentiality, integrity, and availability of the data they collect and transmit.

As these wireless sensors collect and share data, they ensure the security and privacy of transmitted information becomes critical. In recent years, with an increasing emphasis on security, there has been a growing interest in Multi-Party Computation (MPC). MPC allows multiple parties to compute a joint function over their inputs while keeping those inputs private. The SPDZ protocol is among the most prominent and influential secure computation protocols. While the initial SPDZ protocol and its successor, SPDZ-2, have shown promising results, there were still challenges related to performance, scalability, and overall security.

This paper presents a newly developed protocol named SD-SPDZ (Sensor Data SPDZ). The proposed protocol is based on MPC SPDZ-2 protocol and proposes changes to increase the performance in the preprocessing phase by implementing a new algorithm for the Beaver triples calculation. This protocol enhances the privacy-preserving attributes and efficiency of its predecessors. SD-SPDZ integrates advanced cryptographic techniques, offering a more robust and scalable solution for secure computations in WSNs. The primary benefits include reduced communication overhead, faster computation times, and improved resistance against various cyberattacks. The integration of SD-SPDZ in WSNs could improve performance sensitively and change the way sensor data is securely processed in sensor networks. It provides a promising pathway to ensure that as technology advances, the integrity and confidentiality of the data in these networks remain uncompromised.

In summary, as WSNs play an increasingly critical role in modern-day applications, the need for advanced high-performance security mechanisms such as the SD-SPDZ protocol becomes more evident. This combination of cutting-edge, high-performance, secure computation with wireless sensor networks promise a future where data can be both globally accessible and privately computed, bridging the gap between performance and privacy.

Keywords

WSN, Information security, sensor data encryption, SPDZ, SD-SPDZ, Fixed Block Ciphers

1. Introduction

Wireless Sensor Networks (WSN) [1] are being used in numerous applications ranging from environmental monitoring to defense and healthcare. The distributed nature of WSNs and their deployment in potentially hostile environments make data encryption crucial to ensure data confidentiality, integrity, and authenticity. Historically, traditional encryption algorithms such as Advanced Encryption Standard (DES) [2] and Data Encryption Standard (DES) [3] were evaluated for WSNs. However, due to resource constraints in WSN nodes, some additional encryption techniques gained popularity.

Constraints and Challenges

Limited Resources: WSN nodes typically have limited processing capability, memory, and energy.
Dynamic Network Topology: Nodes can join or leave, posing challenges for key management.
Physical Vulnerability: Sensor nodes may be deployed in hostile environments, susceptible to physical attacks.

Current Encryption Techniques

Lightweight Block Ciphers: They require less computational power and memory [4].
Stream Ciphers: Focus on processing data bit-by-bit, requiring minimal memory [5]. Examples are Trivium and Grain.
Public Key Cryptography: Though resource-intensive, they can be optimized for specific tasks like initial key exchange [6].
Multi-Party Computation: Multi-Party Computation (MPC) [7] is a subfield of cryptography that enables multiple parties to jointly compute a function over their inputs

BISEC'23: 14th International Conference on Business Information Security, November 24, 2023, Niš, Serbia
*Corresponding author.
akalexandrov@ir.bas.bg (A. K. A.)

without revealing those inputs to each other.

The main benefits of the MPC based encryption protocols are:

Privacy: Ensures that individual inputs remain secret from other participants.

Correctness: Guarantees that the output is correct even if some participants behave maliciously.

This essential in some WSN's as:

Secure voting systems where voters want to compute the result without revealing individual votes;

Military applications;

Collaborative data analysis in medical research where institutions want to compute a joint result without sharing patient data directly.

1.1. Sensor data encryption techniques

With the rising proliferation of the Internet of Things (IoT) and the widespread deployment of sensor networks across various industries, ensuring the confidentiality, authenticity, and integrity of sensor data has become paramount. This study delves deep into the techniques and strategies employed for sensor data encryption, focusing on the unique challenges and requirements presented by these environments.

Objectives

To understand the peculiarities and constraints of sensor data. To evaluate existing encryption methodologies suitable for sensor data. To propose efficient techniques or improvements tailored for sensor data encryption.

Characteristics of Sensor Data

Sensor data can be distinguished by:

- High volume: Many sensors generate data continuously.
- Temporal relevance: Some data may be time-sensitive.
- Varying importance: Not all sensor data is equally critical.

Challenges in Sensor Data Encryption

- Resource Limitations: Sensors often have constrained processing capabilities, energy, and memory.
- Transmission Overheads: Encryption might introduce additional latency or payload.
- Diverse Deployment: Sensors can be found in hostile environments, making them susceptible to physical attacks.

2. Related works

In the area of the existing approaches, protocols, and algorithms used to reduce the encrypted communication overhead in WSNs the following is commonly used nowadays: BGW Protocol: The Beimel, Malkin, and Micali (BGW) protocol [8] is one of the foundational works in the area of secure multi-party computation. SPDZ can be viewed as a descendant of the BGW protocol, where both focus on achieving security against a malicious adversary.

TinyOT: An efficient protocol [9] for two-party computation, TinyOT inspired many techniques used in SPDZ, especially the ones in the preprocessing phase. Overdrive2K: Overdrive refers to optimizations and enhancements of the SPDZ protocol, further improving the efficiency of the offline phase [10].

MASCOT: A follow-up to SPDZ, MASCOT introduces a more efficient method [11] for the preprocessing phase by using oblivious transfer instead of somewhat homomorphic encryption, reducing computational overhead.

SPDZ2k: The SPDZ2k protocol [12] has been adjusted to operate with calculations based on powers of two.

The significant difficulty with this is that in Z2k, not every component has an inverse, an essential factor for ensuring the security of both MASCOT and SPDZ. To address this, SPDZ2k shifts to Z2k', where k' is a greater value, to offset the presence of zero divisors.

MP-SPDZ: provides a complete implementation of SPDZ2k [13] and features its distinct Z2k version, which is optimized for compile-time k.SPDZ-2: An optimized version of the original SPDZ, it enhances the online phase for better efficiency.

BMR. Beaver and colleagues introduced a method [14] to create garbled circuits from any multi-party computation framework while maintaining security attributes. This method was later enhanced by Lindell and team by employing SPDZ as the foundational protocol. MP-SPDZ integrates BMR with the SPDZ/MASCOT protocol and other security model protocols. Even though this feature wasn't included in SPDZ-2, it was unveiled partially prior to MP-SPDZ's first edition, as it was utilized by Keller and Yanai in their oblivious RAM development.

Yao's Garbled Circuits. Bellare and co-authors showcased a version of Yao's garbled circuits optimized for DES-NI, which is the standard DES execution on contemporary processors [15]. After the final release of SPDZ-2, this version was incorporated and recently updated to encompass the half-gate method.

2.1. SPDZ and SPDZ-2 Encryption Protocols Overview

The SPDZ protocol is a foundational Multi-Party Computation (MPC) scheme known for its robust security

guarantees and practical efficiency. SPDZ facilitates secure computation among multiple parties as connected sensor modules, ensuring that individual inputs remain private.

Protocol Basics

At a high level, the SPDZ protocol encompasses two main phases: **Preprocessing Phase**: Offline phase where correlated randomness (like Beaver Triples) is generated without knowing the inputs.

Online Phase: Actual computation is performed using the preprocessed data.

Secret Sharing in SPDZ

Given a secret s, it is split into additive shares s_1, s_2, s_3, $s_4 \ldots, s_n$ such that:

$$s = \sum_{i=1}^{n} s_i. \tag{1}$$

In the preprocessing phase, a Beaver's triples (a, b, c) are generated where $c = a \times b$. During the online phase, given shares of values x and y that need to be multiplied, the protocol proceeds as:

Compute

$$\delta_x = x - a \tag{2}$$

and

$$\delta_y = y - b. \tag{3}$$

Each sensor module locally computes

$$x \times y = x + \delta_x \times b + \delta_y \times a + \delta_x \times \delta_y \tag{4}$$

In the online phase both values x and y where

$$x = \sum_{i=1}^{n} x_i, \tag{5}$$

$$y = \sum_{i=1}^{n} y_i \tag{6}$$

are computed as:

$$x + y = \sum_{i=1}^{n} (x_i + y_i) \tag{7}$$

Each sensor module locally adds its shares. Using Beaver's triple, multiplication can be securely performed as outlined above.

The SPDZ protocol also integrates zero-knowledge proofs to ensure correctness without revealing individual inputs or intermediate results.

Mathematically, SPDZ employs techniques from linear secret-sharing schemes to ensure zero-knowledge properties.

Basics of the SPDZ-2 Protocol

The SPDZ-2 protocol [16] is an improvement over the original SPDZ protocol for secure multi-party computation (MPC). It builds upon the foundations of the original protocol while addressing certain performance and security issues. The SPDZ-2 protocol also employs two main phases like its predecessor:

Preprocessing Phase: Where correlated randomness is generated.

Online Phase: Where the actual computation using the preprocessed data takes place.

SPDZ-2 introduces a more efficient zero-knowledge proof system to ensure that:

- The shares of each party are consistent.
- The Beaver's triples are valid.

Instead of employing full-fledged zero-knowledge proofs, SPDZ-2 uses MACs (Message Authentication Codes) and correlated randomness to ensure honesty and correctness without much communication overhead.

Improvements over the original SPDZ

Reduced Communication Overhead: By leveraging MACs and efficient consistency checks, SPDZ-2 reduces the number of rounds of communication, which is especially beneficial in settings with many parties. To ensure consistency of shares and validity of the triples, MACs (Message Authentication Codes) are utilized.

The preprocessing phase is made more efficient, leading to faster overall computation times. At the same time, when applied to wireless sensor networks, the SPDZ-2 protocol can still exhibit considerable communication overhead. Sensor networks have bandwidth constraints, limited battery life, and operate in high-latency environments, making communication efficiency crucial.

SPDZ-2 Protocol implementation in Wireless Sensor Networks (WSN)

Wireless Sensor Networks (WSN) typically consist of spatially distributed autonomous devices that cooperatively monitor physical or environmental conditions.

Applying the SPDZ-2 protocol in WSN enables secure collaborative data processing without revealing individual sensor readings.

For a WSN with n sensor nodes, let each node i have a private value v_i. The goal is to compute a function $f(v_1, v_2, \ldots, v_n)$ securely.

Secret sharing in WSN

A sensor node's private value v_i is split into additive secret shares distributed among other nodes such that:

$$v_i = \sum_{i=1}^{n} share_{ij} \tag{8}$$

For shared values x and y, use preprocessed triples (a, b, c) where $c = a \times b$.

Calculate and open

$$\delta_x = x - a, \tag{9}$$

and

$$\delta_y = y - b, \tag{10}$$

to all nodes. Each node locally computes

$$x \times y = c + \delta_x \times b + \delta_y \times a + \delta_x \times \delta_y. \tag{11}$$

Zero-Knowledge Proofs

To ensure consistency of shares and validity of the triples, MACs (Message Authentication Codes) [17] are utilized. Given a MAC key α, and a value v, the MAC is:

$$MAC_v = \alpha \times v. \tag{12}$$

Sensor nodes verify the validity of MACs without revealing their private values.

Communication Model in WSN

Given the energy and bandwidth constraints in WSN, the application of SPDZ-2 requires efficient communication models, possibly hierarchical or cluster-based, to minimize overhead.

In WSN, sensor nodes can be viewed as parties in the MPC. Each node can hold a piece of the secret (i.e., its measurement) and wants to perform computations without revealing its exact measurement to others.

Sensor Data Aggregation

For an aggregate function f over sensor data $d_1, d_2, \ldots, d_n$:

$$f(d_1, d_2, \ldots, d_n) = \sum_{i=1}^{n} f(d_i). \tag{13}$$

Using SPDZ-2, the function f can be computed in a distributed manner without revealing individual d_i values.

Challenges and Solutions in WSN

Bandwidth Constraint

Solution: Use compact secret sharing schemes and optimize communication patterns, possibly adopting hierarchical sensor node structures where cluster heads manage intra-cluster communication.

Energy Constraint

Solution: Minimize interactive rounds in the protocol and consider energy-efficient cryptographic operations. Asynchronous operations can be adapted to allow nodes to enter low-energy states when not actively participating.

Node Failures

Solution: Employ error-correcting codes for share recovery and design the protocol to be resilient to node dropouts.

Security Considerations

In WSN, the threat model may differ, with concerns of node capture or eavesdropping. The security of SPDZ-2 in such a model ensures:

- Privacy: Individual sensor readings are kept confidential.
- Integrity: The outcome of the computation is correct even if some nodes are malicious.

3. Case study

3.1. Sensor Data Communication Overhead in the SPDZ-2 Protocol

The SPDZ-2 protocol, when applied to sensor networks, still has a significant communication overhead. This is especially problematic for wireless sensor networks, which may have limited bandwidth or be subjected to high-latency communication environments.

Communication Overhead in SPDZ

The communication overhead in the SPDZ protocol primarily arises from:

- Calculation, sharing and, reconstructing values in the preprocessing phase.
- Exchanging values during the online phase for operations like multiplication using Beaver's triples.
- Zero-knowledge proofs ensure honesty and correctness.

Strategies to Reduce Communication Overhead

Before initiating the SPDZ protocol, sensors can locally aggregate or summarize their data. For instance, instead of sending individual readings, sensors can send averages or other statistical summaries over a time window.

Group multiple operations together, especially during the preprocessing phase. This can help amortize the cost of generating and distributing values like Beaver's triples over multiple operations.

Instead of running individual proofs for each operation, consider batched or aggregated proofs that can cover multiple operations at once.

Implement secret sharing schemes that are tailored for sensor networks. These can focus on minimizing the number of shares or using techniques like error-correcting codes to handle lost or delayed shares without retransmission.

Employ data compression algorithms to reduce the size of the transmitted data. This can be especially effective if sensor readings or intermediate values in the SPDZ protocol have redundancy or predictable patterns.

Instead of all-to-all communication, consider using relay nodes or hierarchical structures where a subset of sensors aggregates data and communicates with other groups, reducing the total communication across the network.

Instead of continuous computation, synchronize the computation in intervals. This allows for more batched operations and fewer real-time communication requirements. Reducing the communication overhead in the SPDZ protocol when applied to sensor networks requires a combination of algorithmic optimizations, architectural considerations, and leveraging domain-specific knowledge of sensor data. Implementing the above strategies can significantly enhance the efficiency of the SPDZ protocol in sensor environments.

The current paper focuses on the algorithms related to reducing the communication overhead in the preprocessing phase of the SPZD-2 protocol. One of the possible ways to reduce the communication overhead in the preprocessing phase of the SPDZ protocol in WSNs is to use technique such Fixed-key block ciphers.

Fixed-key block ciphers [18], as the name suggests, involve the use of block ciphers with a fixed, predefined key. The idea behind using a fixed key is to transform the block cipher into a deterministic function with pseudorandom behavior.

Standard Block Cipher: A standard block cipher can be denoted as:

$$E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n \tag{14}$$

where E is the encryption function. The first parameter is a key of length k bits. The second parameter is a plaintext block of length n bits. The output is a ciphertext block of length n bits. For a given key K and plaintext P, the encryption is denoted as

$$C = E\left(K, P\right) \tag{15}$$

Fixed-Key Block Cipher: When we talk about a fixed-key block cipher, the key remains constant. This can be represented as:

$$E_{K_{fixed}} : \{0,1\}^n \rightarrow \{0,1\}^n \tag{16}$$

where K_{fixed} is a predefined constant key. For any input block P, the output is $E\left(K_{fixed}, P\right)$.

With the key fixed, a block cipher behaves like a pseudorandom permutation (PRP) over the set of n-bit strings. This means that for every input P, there is a unique output C, and the relationship appears random unless you know the fixed key.

```
Function FixedKey_DDESES_Encrypt(input_block):
// Define a fixed key; this remains constant.
    FIXED_KEY = "32-byte key derived
    from a secure process"

// Use DES encryption with the fixed key.
    ciphertext = DES_Encrypt(FIXED_KEY,
    input_block)
        return ciphertext
End Function

Function FixedKey_DES_Decrypt(ciphertext):
// Define the same fixed key.
    FIXED_KEY = "32-byte key derived from
    a secure process"
// Use DES decryption with the fixed key.
    plaintext = DES_Decrypt(FIXED_KEY,
    ciphertext)
        return plaintext
End Function
```

The `FIXED_KEY` should be securely generated, preferably using a cryptographically secure random number generator, and then kept constant for all future operations. Storing cryptographic keys securely is essential. Depending on the application, you might consider using hardware security modules, secure key storage services, or other best practices.

It is essential to ensure that the `input_block` has an appropriate size for the block cipher is used. For DES, this would typically be 128 bits (or 16 bytes). For the same input, the output will always be the same since the key remains constant.

Since block ciphers are permutations for a given key, the process is reversible. If you know the fixed key, you can decrypt any ciphertext produced by the fixed-key block cipher to retrieve the original input.

In the context of secure multi-party computation (SMPC), fixed-key block ciphers can be used to produce correlated randomness between parties or derive other types of structured randomness efficiently.

One notable application is in the generation of "oblivious pseudorandom functions" (OPRFs) where one party learns the output of a PRF on a specific input without the other party learning anything about the input or the output.

Integration between Beaver triple and Fixed-Key Block Ciphers

Beaver triples and fixed-key block ciphers are both techniques used within the realm of secure multi-party computation (SMPC). While they serve different primary functions and can sometimes be complementary, they can also be seen as alternative techniques in specific settings.

Primarily used for securely computing multiplication in SMPC protocols, Beaver triples [19] consist of preprocessed random multiplicative triples (a,b,c) where c=a×b. These triples allow parties to perform multiplication on secret-shared values without revealing their actual inputs.

The generation of Beaver triples can be computationally intensive, especially in protocols that require a large number of such triples. However, once generated, they make the online phase of the computation faster. Used widely in SMPC protocols like SPDZ and its variants. They are fundamental for protocols that rely on secret sharing and require multiplication operations.

Beaver Triples offer strong security guarantees when generated correctly. Their security relies on the fact that the triples are random and independent of the inputs on which they will be used.

Fixed-Key Block Ciphers: Used to generate certain types of correlated randomness in SMPC. A fixed-key block cipher is a pseudo-random function where the key remains constant. Given the same input, it will always produce the same output, but changing even one bit of the input will produce a substantially different output.

Typically, block ciphers are relatively efficient, especially in hardware implementations. Using them to produce correlated randomness can sometimes be more efficient than generating Beaver triples, depending on the protocol and context. Often used in oblivious pseudo-random function (OPRF) [20] contexts and other settings where correlated randomness or specific patterns of randomness are required.

The security here typically depends on the underlying block cipher's robustness and resistance against cryptographic attacks. If a cryptographically secure block cipher is used, the fixed-key variant can provide strong security guarantees for its purpose.

3.2. Reducing the Sensor Data Communication Overhead in the SD-SPDZ Protocol

Utilizing fixed-key block ciphers to substitute the Beaver triple generation in the SPDZ preprocessing phase is an advanced topic in secure multi-party computation, and this approach is at the core of the new proposed SD-SPDZ protocol.

The idea behind this technique is to use block ciphers, like DES, to deterministically generate shared randomness, which can be used to produce Beaver triples.

The high-level approach for this is:

Key Generation: Each party selects a secret key for the block cipher (e.g., DES).

Beaver triple generation using Fixed-Key Block Ciphers:

Generation of a: Each party P_i generates a random value. Each party computes:

$$A_i = Encrypt_{key_i}(a_i) \tag{17}$$

and broadcast it. The shared value a is the sum of the a_i values.

Generation of b: Each party P_i generates a random value b_i. Each party computes:

$$B_i = Encrypt_{key_i}(b_i) \tag{18}$$

and broadcast it. The shared value b is the sum of the b_i values.

Generation of c: The shared value $c = a \times b$ is computed. However, instead of interacting to verify the correctness of this multiplication, the sensor modules can use the fact that they have encryption of the values a_i and b_i. They can derive the product of the encrypted values, given the properties of the fixed-key block cipher and the determinism of their chosen function. This step avoids the need for complex interactive proofs, hence removing the original need for Beaver triples.

```
function generate_triples_using_block_cipher():
    # a-values
    a_i = random_value()
  A_i = Encrypt_with_fixed_key(key_i, a_i)
    broadcast(A_i)
    a = sum_of_broadcasted_A_values
    # b-values
    b_i = random_value()
  B_i = Encrypt_with_fixed_key(key_i, b_i)
    broadcast(B_i)
    b = sum_of_broadcasted_B_values
    # Compute c using encrypted values and
    # properties of the block cipher
    c= compute_all_A_values, all_B_values)
    return (a, b, c)
```

This approach dramatically simplifies the preprocessing phase compared to the standard SPDZ protocol with Beaver triples and reduces the sensor data communication overhead. However, it assumes that the fixed-key block cipher has certain properties that make this method secure and that the encryption/decryption operations are performed in a secure manner.

Lab environment

The lab environment consists of a cluster-based sensor network consisting of five sensor modules based on NUCs Gigabyte and control center shown in the picture below:

The testing software is implemented in each sensor module and at the cluster head (CH). The experimental results are shown in the table below which describes the average time in seconds to compute 10.000 triples in a WSN cluster consisting of five sensor nodes:

Table 1
Experimental results

MPC protocol Preprocessing phase	Standard Beaver Triple calculation	Fixed-Key Block Ciphers triple calculation
SPDZ	7	-
SPDZ-2	4	-
SD-SPDZ	4	0.7

Figure 1: Cluster-based sensor network consisting of five sensor modules based on NUCs Gigabyte and control center shown in the picture below.

4. Conclusion

This paper presents a newly developed protocol named SD-SPDZ (Sensor Data SPDZ). The proposed protocol is based on MPC SPDZ-2 protocol and proposes changes to increase the performance in the preprocessing phase by implementing a new algorithm for the Beaver triples calculation.

This protocol enhances the privacy-preserving attributes and efficiency of its predecessors. SD-SPDZ integrates advanced cryptographic techniques, offering a more robust and scalable solution for secure computations in WSNs. The primary benefits include reduced communication overhead, faster computation times, and improved resistance against various cyberattacks.

The integration of SD-SPDZ in WSNs could improve performance sensitively and change the way sensor data is securely processed in sensor networks. It provides a promising pathway to ensure that as technology advances, the integrity and confidentiality of the data in these networks remain uncompromised.

In summary, as WSNs play an increasingly critical role in modern-day applications, the need for advanced high-performance security mechanisms such as the SD-SPDZ protocol becomes more evident. This combination of cutting-edge, high-performance, secure computation with wireless sensor networks promises a future where data can be both globally accessible and privately computed, bridging the gap between performance and privacy.

References

[1] Y. Pinar, A. Zuhair, A. Hamad, A. Resit, K. Shiva, A. Omar, Wireless sensor networks (WSNs), in: 2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT), IEEE, 2016, pp. 1–8.

[2] J. Zhao, Des-co-rsa: A hybrid encryption algorithm based on DES and RSA, in: 2023 IEEE 3rd International Conference on Power, Electronics and Computer Applications (ICPECA), IEEE, 2023, pp. 846–850.

[3] N. Ahmad, S. R. Hasan, A new asic implementation of an advanced encryption standard (AES) crypto-hardware accelerator, Microelectronics Journal 117 (2021) 105255.

[4] Y. Li, J. Feng, Q. Zhao, Y. Wei, Hdlbc: A lightweight block cipher with high diffusion, Integration 94 (2024) 102090.

[5] H. Noura, O. Salman, R. Couturier, A. Chehab, Lesca: Lightweight stream cipher algorithm for emerging systems, Ad Hoc Networks 138 (2023) 102999.

[6] K. Pavani, P. Sriramya, Enhancing public key cryptography using RSA, RSA-CRT and N-prime RSA with multiple keys, in: 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), IEEE, 2021, pp. 1–6.

[7] H. Goyal, S. Saha, Multi-party computation in iot for privacy-preservation, in: 2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS), IEEE, 2022, pp. 1280–1281.
[8] R. Gennaro, M. Di Raimondo, Secure multiplication of shared secrets in the exponent, Information processing letters 96 (2005) 71–79.
[9] C. Hazay, P. Scholl, E. Soria-Vazquez, Low cost constant round MPC combining bmr and oblivious transfer, Journal of cryptology 33 (2020) 1732–1786.
[10] E. Orsini, N. P. Smart, F. Vercauteren, Overdrive2k: efficient secure MPC over from somewhat homomorphic encryption, in: Cryptographers' Track at the RSA Conference, Springer, 2020, pp. 254–283.
[11] I. Damgård, V. Pastro, N. Smart, S. Zakarias, Multiparty computation from somewhat homomorphic encryption, in: Annual Cryptology Conference, Springer, 2012, pp. 643–662.
[12] R. Cramer, I. Damgård, D. Escudero, P. Scholl, C. Xing, SPDZ2k: efficient MPC mod 2k for dishonest majority, CRYPTO, 2018.
[13] M. Keller, Mp-spdz: A versatile framework for multi-party computation, in: Proceedings of the 2020 ACM SIGSAC conference on computer and communications security, 2020, pp. 1575–1590.
[14] M. Bottarelli, P. Karadimas, G. Epiphaniou, D. K. B. Ismail, C. Maple, Adaptive and optimum secret key establishment for secure vehicular communications, IEEE Transactions on Vehicular Technology 70 (2021) 2310–2321.
[15] H.-J. Kim, H.-I. Kim, J.-W. Chang, A privacy-preserving kNN classification algorithm using Yao's garbled circuit on cloud computing, in: 2017 IEEE 10th international conference on cloud computing (CLOUD), IEEE, 2017, pp. 766–769.
[16] J. Liu, Y. Tian, Y. Zhou, Y. Xiao, N. Ansari, Privacy preserving distributed data mining based on secure multi-party computation, Computer Communications 153 (2020) 208–216.
[17] G. Arumugam, V. L. Praba, S. Radhakrishnan, Study of chaos functions for their suitability in generating message authentication codes, Applied Soft Computing 7 (2007) 1064–1071.
[18] C. Guo, J. Katz, X. Wang, Y. Yu, Efficient and secure multiparty computation from fixed-key block ciphers, in: 2020 IEEE Symposium on Security and Privacy (SP), IEEE, 2020, pp. 825–841.
[19] J. B. Nielsen, P. S. Nordholt, C. Orlandi, S. S. Burra, A new approach to practical active-secure two-party computation, in: Annual Cryptology Conference, Springer, 2012, pp. 681–700.
[20] S. Casacuberta, J. Hesse, A. Lehmann, SoK: Oblivious pseudorandom functions, in: 2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P), IEEE, 2022, pp. 625–646.