International Journal of Future Engineering Innovations

Survey of verification communication protocol security using computational models

Sarah Basim Abed [1] and Ghadeer Ibrahim Maki [2]
[1] Department of Accounting Techniques, Al-furat Al-awsat Technical University, Technical Institute / Diwaniyah, Najaf, Iraq
[2] Department Social Health, Al-furat Al-awsat Technical University, Technical Institute / Diwaniyah, Najaf, Iraq

* Corresponding Author: **Sarah Basim Abed**

Article Info

ISSN (online): 3049-1215
Volume: 02
Issue: 03
May-June 2025
Received: 01-03-2025
Accepted: 03-04-2025
Page No: 38-45

Abstract
With the rapid growth of technologies like the Internet of Things (IoT) and social media, securing communication channels against unauthorized access has become a critical priority. Protecting privacy and ensuring secure interactions requires protocols that not only encrypt data during transmission but also verify the identities of all parties involved. This paper examines widely-used computational models that validate the security of such protocols. It provides an overview of these models, their types, and their roles in analyzing and achieving mutual authentication and secure key exchange. By understanding these frameworks, we can better assess how communication protocols resist attacks and maintain confidentiality in dynamic environments.

DOI: https://doi.org/10.54660/IJFEI.2025.2.3.38-45

Keywords: Protocols, computational models, authenticated key exchange, random oracle model, secure channels

1. Introduction

Protocols are established guidelines that enable two or more parties to exchange data securely and accurately [1]. These protocols aim to achieve two core security objectives:

1. Authentication: Ensuring all parties can verify each other's identities (e.g., using passwords, digital certificates, or biometric methods) [2].
2. Secure Key Exchange: Generating a shared secret key to encrypt messages before transmission and decrypt them upon receipt. By combining these elements, protocols prevent unauthorized access and tampering, creating a trusted environment for communication.

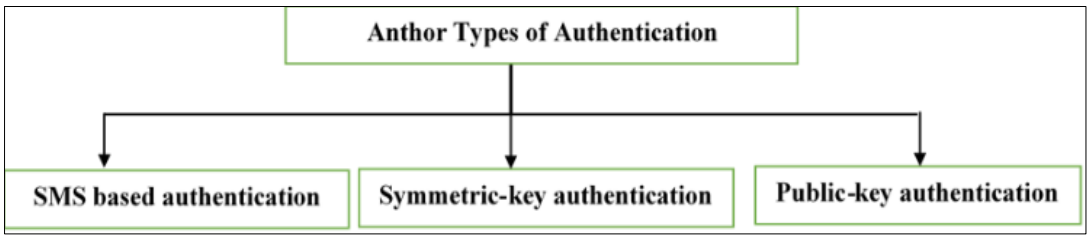

Fig 1: Types of Authentication

While protocols can establish secure communication sessions between parties, a major challenge arises from adversaries who may attempt to disrupt or manipulate these interactions [2]. These attackers can alter messages (e.g., deleting, adding, or modifying data), impersonate legitimate users, or even hijack sessions entirely. To counter such threats, protocols must be rigorously designed to demonstrate their resilience against attacks. This involves two critical steps:

1. Understanding the Adversary: Identifying their goals (e.g., stealing data, disrupting communication) and the extent of their capabilities (e.g., limited vs. unlimited access to resources).

2. Choosing the Right Security Model: Selecting frameworks that align with the protocol's design to validate its security claims [3].

Adversaries vary in their ability to corrupt systems, ranging from basic tampering to advanced impersonation tactics. Classifying these threats based on their corruption methods helps tailor defenses effectively.

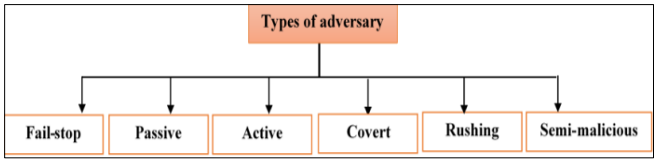

Fig 2: Types of Adversary

To defend against attackers, protocols must be designed to restrict their capabilities. One approach is to use mathematical models-like the random oracle-to test whether a protocol meets its security goals [4]. However, there are no universal standards to evaluate these models. Ideally, models should be self-contained, avoiding reliance on external assumptions or undefined parameters [3]. Clarity is key: models must be precise and unambiguous to prevent misinterpretation. This ensures protocols can operate independently without interdependencies. In this paper, we explore security models tailored for key exchange, whether between two parties, three parties, or larger groups.

2. Random oracle model

To assess how an attacker might compromise a protocol's security—such as stealing encryption keys—we first define their capabilities. Attackers can probe the protocol by sending queries (requests for information) to an oracle, a simulated tool representing their control over all network communications. In this model, attackers can intercept or manipulate any session, sending queries to the oracle and receiving responses that mimic real-world interactions. However, these queries must follow predefined rules accepted by the oracle [4].

The system operates like a secure, self-contained tool. When it receives a valid input (query), it provides the correct response. For unrecognized inputs, it generates a random response, saves it for future use, and ensures the same random answer is given if the same input is repeated. This behavior can be summarized as follows [5]:

1. **New Input:** If the system encounters an unfamiliar query, it creates a random response from its possible outputs, saves both the query and response, and returns the result.
2. **Known Input:** If the query has been processed before, the system retrieves and returns the exact response it saved earlier.

The Random Oracle Model (ROM) was first introduced by Bellare and Rogaway [6]. Unlike standard hash functions, the ROM ensures that outputs are unpredictable—even to attackers or legitimate participants. For instance, if an attacker sends a request through a secure channel, the oracle internally computes a response and securely returns it. While ROM is widely used to validate protocol security, it has limitations: real-world hash functions cannot fully mimic its idealized randomness, meaning security guarantees may weaken in practice. Despite this, ROM remains a preferred framework for security analysis due to its practicality and effectiveness compared to other models [7].

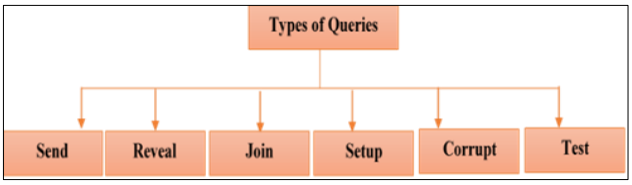

Fig 3: Types of Queries

Queries	Definition
Send	When an attacker attempts to join a secure session without valid credentials, they begin by sending a "Send" request. The system evaluates this request against its security rules. Based on this check, the system can either: - Accept the attacker as a participant (without granting secret information), - Reject the request outright, or - Return a response message for further action.
Join	During the protocol's execution, this type of request is used to simulate eavesdropping. It is reserved for attackers, as legitimate participants in the session do not initiate such actions, a simulated version of the attacker's behavior is created.
Setup	During the protocol's execution, a specialized tool "query" is used to mimic eavesdropping attacks. This tool is reserved for simulating how an attacker might intercept communications, as legitimate participants in the session do not engage in such activities. To rigorously test the protocol's defenses, researchers create a simulated version of the attacker's methods, allowing them to assess how effectively the protocol detects and blocks unauthorized interception attempts
Corrupt	This method allows attackers to steal long-term encryption keys (used for securing data over extended periods) and take control of communication sessions by pretending to be a legitimate participant. By impersonating a trusted user.
Test	The "Test Query" is a security challenge that allows an attacker to check if they can guess the encryption keys used in a session. Here's how it works: 1. The Challenge: A security system (Oracle) generates a random value-either a 0 or a 1. - If the value is 1, the attacker is given the real encryption keys. - If the value is 0, the attacker receives a fake, random string instead. 2. One Attempt Only: The attacker can perform this test once

To ensure the security of protocols and evaluate their performance, various models are utilized. In this discussion, we will explore these models. The chart that follows categorizes them by their families and types to offer a clear and organized perspective.

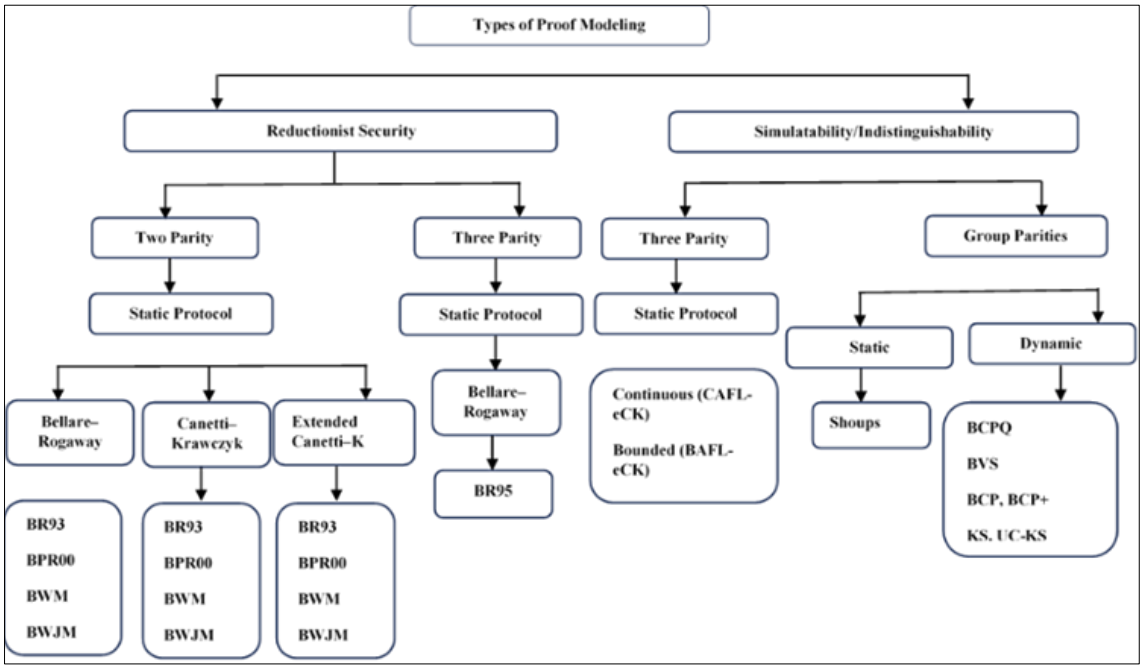

Fig 4: Types of modeling proof security

3. Overview of models

What are the models and why do we use them? Models are functions that are applied to protocols that assume an environment contains both session subscribers and the attacker. They give the attacker a variety of different information depending on the model from the session to see how well the protocol can resist the attacker. These models are used to analyze and break protocols [8].

The information varies from one model to another. There is information that enables the attacker to know the session key, and information that is weak. Here the measure is the quality of the information, not the quantity [9, 10].

A) The models are divided into two main parts: Reductionist Security Proofs and Proofs Based on Simulatability/Indistinguish ability.

1. Reductionist Security Proofs

To test the security of a protocol, researchers use simulated scenarios called security games. These games mimic real-world battles between an attacker and the system. Here's how they work in simple terms:

1. The game begins with publicly shared keys (like digital passwords) and predefined rules.
2. The attacker tries to break the system—for example, by guessing keys or altering messages.
3. The game stops when the attacker either gives up or meets a specific goal (e.g., stealing data within a time limit).
4. The protocol's security is measured by how well it thwarts the attacker. Secure.

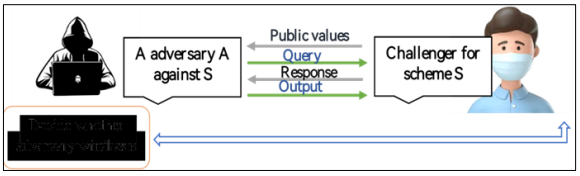

Fig 5: security game

2. Proofs Based on Simulatability/Indistinguishability

Can be considered as a particular kind of computational security strategy, where the security of a cryptographic construction is derived from the indistinguishability of the ideal and real executions.

- Type of modeling
- model by Belair-Roguay (BR)
- Bellare and Rogaway introduced the first security model that formally defined attacker capabilities and established clear security criteria. This groundbreaking work shifted the focus of cryptography research toward practical, attack-resistant protocols.leading to the development of numerous protocols and computational models tailored to counter specific threats [6, 7, 11].

Table 1: Descript Of Br93 Model

Definition	Actions
BR93 model	In 1993, Bellare and Rogaway laid the groundwork for modern entity authentication and secure key exchange protocols. Their BR93 model introduced the idea of "matching conversations" to ensure two parties in a communication (like a client and server) stay synchronized. - Each interaction between the parties is broken down into small, well-defined steps. - A "conversation" is like a timestamped log of these steps, where each entry records: - When the step happened (time p), - What message was received (input q), - How the system responded (output y). $C_A = (\tau 0\ ,\ 'start',\ \alpha 1),\ (\tau 2\ ,\ \beta 1\ ,\ \alpha 2)$

	$C_A = (\tau 0\ ,\ 'start',\ \alpha 1),\ (\tau 2\ ,\ \beta 1\ ,\ \alpha 2)$ and $C_B = (\tau 1\ ,\ \alpha 1\ ,\ \beta 2),\ (\tau 3\ ,\ \alpha 2\ ,\ *\ ),\ for\ \tau 0\ <\ \tau 1\ <\cdots$ $C_B = (\tau 1\ ,\ \alpha 1\ ,\ \beta 2),\ (\tau 3\ ,\ \alpha 2\ ,\ *\ ),\ for\ \tau 0\ <\ \tau 1\ <\cdots$
Properties that determine the security	Mutual identity confirmation and secure key sharing
Drawbacks	1. Each participant in the system has a permanent cryptographic key that they securely share with every other participant. 2. Key exchange and identity verification (authentication) are tightly linked, ensuring both processes happen securely and simultaneously. 3. The system restricts attackers' abilities, limiting how much they can interfere with or compromise security.

Table 2: Descript of BR models

Model	Setting	Partnering mechanism	Gap
BR93	shared Two-party key	Conversation matching	Link authentication to key exchange
BR95	Server dependent	Partner's job	Unpair previous model, but no authentication for reason of using partner function
BPR2000	Password-dependent	Session identifiers	Using a password for authentication is a weak method and therefore does not provide forward data confidentiality.
BWM	Public key	Matching Conversations	It's a good model to photograph and post in a signature.
BWJM	Key agreement	Matching Conversations	model is a good for relying on AK and AKC.

2. Canetti–Krawczyk (CK)

The second family of models includes

Table 3: Descript Of Ck Models

Model	Setting	Partnering mechanism	Gap
BCK98	Two-party shared key	Authenticated links (AM). Unauthenticated links (UM)	When an attacker uses specialized attacks (corruption queries) to steal both temporary session keys and permanent cryptographic keys, Shoup labeled this aggressive tactic as "strong adaptive corruption"
CK01	Two-party shared key	Session identifiers	1. Session IDs: There's no standardized or clear method for generating unique identifiers for communication sessions. 2. Session Status Checks: The model lacks a precise way to inspect or define the current "state" of a session. 3. Query Restrictions: If a session is already in progress, the system blocks new requests to analyze or interact with it until the current session ends.
HMQV	Two-party shared key	Session identifiers	Participant B and their partner C use a unique identifier to track their communication session. This identifier includes: - IDB: Participant B's identifier. - IDC: Participant C's identifier. - MO: Messages received during the session. - MI: Messages sent during the session.

3. Extended CK (eCK)

- It based on The eCK security model relies on two types of secrets to protect communication:
- **Long-term secrets:** Permanent keys (like passwords) shared between participants.
- **Ephemeral secrets:** Temporary keys generated for a single session.

Each session is uniquely identified by

- **Roles:** Each participant has a distinct role.
- **Identifiers:** Unique IDs for both parties (e.g., Alice = IDA, Bob = IDB).
- **Message flow:** Tracks outgoing (Out) and incoming (In) messages during the session.

The model tests security using five types of attacker actions

- **Send:** Intercept or alter messages.
- **Reveal:** Expose session-specific keys.
- **Ephemeral:** Access temporary secrets.
- **Long-term:** Steal permanent keys.
- **Test:** Challenge the system to distinguish real keys from fake ones.

This structure ensures that even if an attacker gains some secrets, the protocol remains secure [12, 13].

Table 4: Descript Of Eck Models

Model	Setting	Powerful
MU08	Two-party shared key	This security model offers limited protection, restricting attackers from performing long-term attacks. While it is less robust than the eCK framework, it provides better defenses compared to other models in certain scenarios.
eCKw	Two-party shared key	An attacker can obtain the long-term key by replaying the message.
eCK-PFS	Two-party shared key	In the long run the opponent gets the key.
seCK	Two-party shared key	it provides the opponent with space to obtain intermediate information on session keys.

The Generic After-the-fact Leakage-eCK ((·) AFL-eCK) Model

- This framework includes two security models designed to handle scenarios where attackers gain access to secret keys after a protocol session ends:
- CAFL-eCK (Continuous After-the-Fact Leakage):
- Allows attackers to obtain specific, limited information from long-term secrets and temporary session keys.
- BAFL-eCK (Bounded After-the-Fact Leakage):
- Permits attackers to gather larger amounts of information continuously over time, including both long-term secrets and session [14].

Table 5: Descript of Models

Continuous After-the-fact Leakage-eCK (CAFL-eCK)	Bounded After-the-fact Leakage-eCK (BAFL-eCK)
In this model, encryption key details might gradually leak during communication between parties. However, the core idea is that an attacker's ability to steal session keys depends entirely on what information they can gather during the interaction	In this model, attackers can only access limited and basic types of information. The total exposure of long-term keys and session initiation attempts is restricted based on the data the attacker manages to gather. The strength of the security hinges on two factors: 1. Quantity: How much information is leaked. 2. Importance: How critical that information is to compromising the system. If the leaked data is highly valuable and sufficient to break the protocol the session is considered breached. However, if the information is partial or insignificant, attackers will find it extremely difficult to infiltrate the session.

Shoup's Simulation Model

Shoup's framework introduces a unique approach to analyzing protocol security by comparing two parallel systems:

- **Ideal System:** Represents a hypothetical, perfectly secure environment where session keys are generated independently and randomly.
- **Real System:** Mimics actual implementations where the adversary controls the network

Three Types of Corruption:

- Static Corruption
- Adaptive Corruption
- Strong Adaptive Corruption
- The table shows the difference between the two systems [15, 10].

Table 6: Descript Of Shoup Model

Real System	Ideal System
1. Attackers control the communication channels—they can eavesdrop, alter, or block messages between participants. 2. However, attackers cannot directly access encryption keys or the random values used to secure sessions. 3. A trusted authority (TTP), similar to a bank managing account security, issues and safeguards long-term keys. Participants request these keys from the TTP to authenticate themselves and establish secure connections.	1. The attacker sets up users and manages sessions by either starting them or leaving them inactive. They also interact with applications as needed during the process. 2. The attacker has full control to decide which communication sessions are established between users. 3. Session keys are not generated through standard methods but are instead created randomly and independently, ensuring no predictable links to other parameters.

Group key exchange Modelling

Protocols consist of two types [16]:

Dynamic-group key-exchange protocols: In order to deal with dynamic group changes in following protocol sessions more efficiently, participants must generally store more auxiliary information.

Dynamic Group Key Exchange (GKE) Protocol Explained:

A dynamic GKE protocol allows a group of users to securely share and update encryption keys, even as members join or leave.

1. Initialization:

- Each member runs a setup process to prepare for secure group communication.

2. Group Setup:

Members use a setup protocol to generate their first shared secret key.

3. Adding Members

When new users join, existing members collaborate with them via a joint protocol to update the group's secret key. This ensures newcomers can't access past messages but can decrypt future ones.

4. Removing Members:

If members leave or are excluded, the remove protocol lets

the remaining users generate a fresh key. This locks out the removed users from future communications.

2. In static protocols, all supporting data (like temporary keys or parameters) is generated fresh each time the protocol runs. This "clean slate" approach reduces risks because no old data is reused. However, dynamic protocols—which retain and update data across sessions—are far more vulnerable to advanced attacks This makes securing dynamic protocols especially critical.

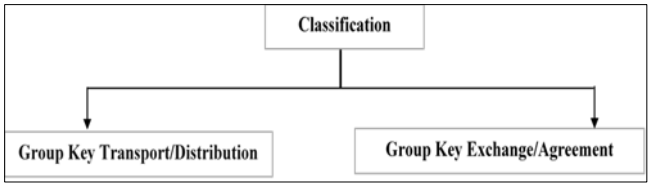

Fig 7: Types of key group

1. Group Key Transport/Distribution

In this type, one of the group members creates the session keys and sends them securely to the rest of the parties participating in the session. The transfer process must be confidential and honest through secret transmission channels. The person responsible for generating this key may be either a person belonging to the group or a trusted third party (TTP) that chooses keys instead of subscribers [17].

2. Group Key Exchange/Agreement

In this type, the shared key is generated through the participation of more than one party in its derivation, so the key information is shared so that the value of these keys cannot be predetermined. Therefore, all members must interact with each other in order to calculate this value [17, 18]. The table 7 shows some of the models used in the analysis of mass protocols [19, 20].

Table 8: Example protocol for each model

Model	Type Key Exchange	Protocol Example
BR93	Two-Party Key Exchange	Diffie and Hellman
BR95	Three-Party Key Exchange	Joux
BPR2000	Two-Party Key Exchange	Diffie and Hellman
BWM	Two-Party Key Exchange	Diffie and Hellman
BWJM	Two-Party Key Exchange	Diffie and Hellman
BCK98	Two-Party Key Exchange	Diffie and Hellman
CK01	Two-Party Key Exchange	Diffie and Hellman
HMQV	Two-Party Key Exchange	Diffie and Hellman
MU08	Two-Party Key Exchange	Diffie and Hellman
eCKw	Two-Party Key Exchange	Diffie and Hellman
eCK-PFS	Two-Party Key Exchange	Diffie and Hellman
seCK	Two-Party Key Exchange	Diffie and Hellman
BAFL-eCK	Two-Party Key Exchange	Diffie and Hellman
CAFL-eCK	Two-Party Key Exchange	Diffie and Hellman
BCPQ	Group Key Exchange	(Burmester and Desmedt), (Steiner, Tsudik, and Waidner)
BCP	Group Key Exchange	(Ateniese, Steiner, and Tsudik), (Barua, Dutta, and Sarkar)
BCP+	Group Key Exchange	Lee, Kim, Kim, and Ryu
KS	Group Key Exchange	Ingemarsson, Tang, and Wong
UC-KS	Group Key Exchange	Ateniese, Steiner, and Tsudik
BVS	Group Key Exchange	Becker and Wille

Table 9: Compression between models from where queries

Models	Send	Reveal	State Reveal	Corrupt	Test
BR93	Yes	Yes	No	Yes	Yes
BR95	Yes	Yes	No	Yes	Yes
BPR2000	Yes	Yes	No	No	Yes
BWM	Yes	Yes	No	Yes	No
BWJM	Yes	Yes	No	Yes	No
BCK98	Yes	Yes	No	Yes	No
CK01	Yes	Yes	Yes	Yes	Yes
HMQV	Yes	Yes	Yes	No	Yes
MU08	Yes	Yes	No	No	Yes
eCKw	Yes	Yes	No	No	Yes
eCK-PFS	Yes	Yes	No	No	Yes
seCK	Yes	Yes	No	No	Yes
BAFL-eCK	Yes	Yes	No	Yes	No
CAFL-eCK	Yes	Yes	No	Yes	No
BCPQ	Yes	Yes	No	Yes	Yes
BCP	Yes	No	No	Yes	Yes
BCP+	Yes	Yes	No	Yes	Yes
KS	Yes	Yes	Yes	Yes	Yes

Table 10: Descript of group key exchange models

Models	Definition	Powerful	Drawbacks
Model by Bresson, Chevassut, Pointcheval, and Quisquater (BCPQ)	The first protocol designed for group key exchange differs from earlier approaches that only involved two parties. It extends framework to support secure communication among multiple participants	The BCPQ model allows attackers to send multiple requests to the security system (Oracle) to test vulnerabilities. However, the model's strength lies in its two core security goals: 1. Authenticated Key Exchange (AKE): Ensuring all parties securely agree on a shared encryption key while verifying each other's identities. 2. Mutual Authentication: Guaranteeing that every participant confirms the legitimacy of others in the group.	1. Impersonation Vulnerability: Some group key exchange (GKE) protocols have flaws that allow attackers to pretend to be legitimate users by exploiting oracle. 2. Limited Participant Issue: When these protocols are used for small groups (like just two people), they rely on hashed user IDs (scrambled versions of identifiers for security). 3. Late-Availability Problem: The security of these protocols depends on analyzing actual interactions ("conversations") between users. But this data isn't available until *after* the protocol is already in use.
Models by Bresson, Chevassut, and Pointcheval (BCP)	This updated version of the BCPQ model enhances its capabilities to handle dynamic group key exchange, where participants can join or leave the group during active communication. The constant changes in group membership disrupt attackers' strategies, making it harder for them to predict or interfere with the secure exchange of keys.	The BCP model uses three core actions to manage secure group communication: - Setup - Join - Remove Like its predecessor (BCPQ), the BCP model ensures two critical security goals: 1. AKE-Security 2. MA-Security	The protocol's defenses against attacks mirror the strategies used in the BCPQ model
Models by Bresson, Chevassut, and Pointcheval (BCP+)	This enhanced model addresses the vulnerabilities of earlier approaches, delivering the highest level of defense against adversarial threats.	1. Hardware-Based Security: - Secure Coprocessor: Acts like a digital vault, storing sensitive session data (e.g., temporary keys, logs) in a tamper-proof environment. - Smart Cards: Function as uncloneable ID badges, securely holding users' long-term keys (permanent cryptographic passwords) to prevent exposure. 2. Forward Secrecy in BCP+ Model: - Weak Forward Secrecy (wfs): Protects past sessions even if attackers later steal keys. However, attackers can still use certain tactics (like intercepting messages or corrupting devices) to target. - Adversary Tools: In this model, attackers can exploit all queries from the original BCP model (e.g., intercepting messages) and additional tactics like Send (altering messages), Sends (spoofing users), and Corrupts (stealing keys).	Expensive in both time and cost
Models by Katz and Shin (UC-KS)	Implementing an "ideal protocol" means creating a system that behaves like a perfectly secure version of itself in theory.	This model uses a unique method to prove security: it checks if an attacker can tell the difference between the real protocol and a hypothetical, perfectly secure version (simulation approach). Other models rely on proving security by linking it to solving a known hard problem (reductionist approach). Katz and Shin created a "translator tool" (compiler) that takes any group key exchange (GKE) protocol proven secure under the BCPQ model and converts it into a protocol that meets stricter security standards (UC-based model). This ensures the protocol stays secure even when combined with other systems, like fitting a puzzle piece into a larger, secure framework.	1. Katz and Shin's model lacks evidence that their "translator tool" (compiler) can block attacks where attackers impersonate legitimate users from within the group. This gap leaves uncertainty about whether their security definitions are sufficient to prevent such threats. Researchers still need to verify if their mathematical framework can support rigorous security proofs for real-world scenarios. 2. The model's security criteria overlook critical issues like: - Key Control - Contributiveness
Model by Bohli, Vasco, and Steinwandt (BVS)	This approach isn't entirely new but builds on the BCPQ model introduced by Katz and Yung. It enhances the framework to create unified security protocols that effectively counter sophisticated and aggressive attacks. The goal is to ensure robust protection even when facing advanced threats, strengthening the original model's defenses.	This security model prioritizes two main objectives: 1. Indistinguishable Session Keys: Attackers should be unable to tell the difference between real session keys (used to encrypt messages) and random data generated by the system. 2. Forward Secrecy: Ensures past communications stay secure even if future keys are compromised. Additionally, the model enforces strong authentication: - Legitimate participants with matching session IDs and keys can verify each other's identities. This blocks imposters from sneaking into the conversation.	1. Restricted Attack Methods: Attackers must use a specific, predefined method (set by the system) to try and crack session keys. They can't invent their own tactics, limiting their ability to exploit weaknesses. 2. Blind to Internal Secrets: Even if attackers breach the system, they can't access its hidden settings or secret parameters. These internal details are shielded, so attackers can't learn how the system truly works. 3. Unclear Key Validation: The system lacks a reliable way to confirm whether a key is genuine (created securely) or fake (tampered with by an attacker). This ambiguity makes it hard to trust the keys used.

7. Conclusion

In this paper, we have referred to protocols, their types and objectives, as well as mutual authentication, and the most prominent methods used for authentication and encryption. Providing a secure connection has become a necessary and urgent need at the present time, so it is necessary to prove the

security of the protocols used in communication before applying them. In this paper, we discussed the oldest and most recent models, especially in protocols that contain a large number of participants in the communication (session). For each model there are advantages and disadvantages that led to the emergence of other models, but through the survey we found that the strongest model is BCP+. It provides powerful information to the attacker but is costly in terms of computation and time. The model must be used according to the protocol produced. For each protocol, there are characteristics, for example, two, three or more parties, and the appropriate model is determined for them.

8. References

1. Choo KKR, Boyd C, Hitchcock Y. Errors in Computational Complexity Proofs for Protocols. In: Advances in Cryptology – ASIACRYPT'05. Lecture Notes in Computer Science, vol 3788. Springer; 2005:624-43.
2. Boyd C, Mathuria A, Stebila D. Protocols for authentication and key establishment. Vol 1. Heidelberg: Springer; 2003.
3. Bellare M, Pointcheval D, Rogaway P. Authenticated key exchange secure against dictionary attacks. In: International conference on the theory and applications of cryptographic techniques. Springer; 2000:139-55.
4. Kala DK, Sharma V. Public Key Encryption Algorithm and the Random Oracle. 2009.
5. Kurtz SA, Mahaney SR, Royer JS. Average dependence and random oracles. University of Arizona, Department of Computer Science; 1991.
6. Boyd C, Choo KKR, Mathuria A. An extension to Bellare and Rogaway (1993) model: resetting compromised long-term keys. In: Australasian Conference on Information Security and Privacy. Springer; 2006:371-82.
7. Choo KKR, Hitchcock Y. Security requirements for key establishment proof models: Revisiting Bellare-Rogaway and Jeong-Katz-Lee protocols. In: Australasian Conference on Information Security and Privacy. Springer; 2005:429-42.
8. Alia MA, Tamimi AA, AL-Allaf ON. Cryptography based authentication methods. In: Proceedings of the World Congress on Engineering and Computer Science. Vol 1; 2014.
9. Shoup V. Sequences of Games: A Tool for Taming Complexity in Security Proofs. Cryptology ePrint Archive, Report 2004/332; 2004. Available from: http://eprint.iacr.org/2004/332.pdf
10. Choo KKR, Boyd C, Hitchcock Y. Examining Indistinguishability-Based Proof Models for Key Establishment Protocols. In: Advances in Cryptology – ASIACRYPT'05. Lecture Notes in Computer Science, vol 3788. Springer; 2005:585-604.
11. Choo KKR. A proof of revised Yahalom protocol in the Bellare and Rogaway (1993) model. Comput J. 2007;50(5):591-601.
12. Cremers C. Examining indistinguishability-based security models for key exchange protocols: the case of CK, CK-HMQV, and eCK. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security; 2011:80-91.
13. Jinyue X, Jiandong W, Liming F, Yongjun R, Shizhu B. Formal Proof of Relative Strengths of Security between ECK2007 Model and other Proof Models for Key Agreement Protocols. Cryptology ePrint Archive; 2008.
14. Alawatugoda J, Stebila D, Boyd C. Modelling after-the-fact leakage for key exchange (full version). In: Proceedings of the 9th ACM symposium on Information, computer and communications security; 2014:207-16.
15. Katz J, Yung M. Scalable Protocols for Authenticated Group Key Exchange. In: Advances in Cryptology - CRYPTO'03. Lecture Notes in Computer Science, vol 2729. Springer; 2003:110-25.
16. Bresson E, Chevassut O, Pointcheval D. Dynamic group Diffie-Hellman key exchange under standard assumptions. In: International conference on the theory and applications of cryptographic techniques. Springer; 2002:321-36.
17. Bresson E, Chevassut O, Pointcheval D. Provably authenticated group Diffie-Hellman key exchange—the dynamic case. In: International Conference on the Theory and Application of Cryptology and Information Security. Springer; 2001:290-309.
18. Zhang F, Chen X. Attack on Two ID-based Authenticated Group Key Agreement Schemes. Cryptology ePrint Archive, Report 2003/259; 2003. Available from: http://eprint.iacr.org/2003/259/
19. Shim K. Cryptanalysis of Al-Riyami-Paterson's Authenticated Three Party Key Agreement Protocols. Cryptology ePrint Archive; 2003. Available from: http://eprint.iacr.org/
20. Zhang F, Chen X. Attack on Two ID-based Authenticated Group Key Agreement Schemes. Cryptology ePrint Archive, Report 2003/259; 2003. Available from: http://eprint.iacr.org/2003/259/