Hindawi
Security and Communication Networks
Volume 2021, Article ID 8273172, 13 pages
https://doi.org/10.1155/2021/8273172

Research Article

ESSM: Formal Analysis Framework for Protocol to Support Algebraic Operations and More Attack Capabilities

Xuyang Miao[1,2] **Chunxiang Gu**,[1,2] **Siqi Lu**[1,2] **and Yanan Shi**[1,2]

[1]*State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China*
[2]*Henan Key Laboratory of Network Cryptography Technology, Zhengzhou 450001, China*

Correspondence should be addressed to Siqi Lu; 080lusiqi@sina.com

Received 1 August 2021; Accepted 20 November 2021; Published 7 December 2021

Academic Editor: Youwen Zhu

The strand space model has been proposed as a formal method for verifying the security goals of cryptographic protocols. However, only encryption and decryption operations and hash functions are currently supported for the semantics of cryptographic primitives. Therefore, we establish the extended strand space model (ESSM) framework to describe algebraic operations and advanced threat models. Based on the ESSM, we add algebraic semantics, including the Abelian group and the XOR operation, and a threat model based on algebraic attacks, key-compromise impersonation attacks, and guess attacks. We implement our model using the automatic analysis tool, Scyther. We demonstrate the effectiveness of our framework by analysing several protocols, in particular a three-factor agreement protocol, with which we can identify new attacks while providing trace proofs.

1. Introduction

In recent years, formal analysis has been widely applied to different types of protocol security analyses [1, 2], including 5G authentication and key agreement (AKA) protocol [3, 4], transport-layer security (TLS) version 1.3 [5, 6], Signal Messaging Protocol [7], Secure Forwarding Protocols [8], and Multifactor Authentication Protocols [9]. Theoretical research on the formal analysis is under way, and great progress has been made in observation equivalencies [10] and equality theory [11].

Automatic analysis of algebraic attributes in security protocols is gaining increasing attention in formal analysis. Among the existing formal symbol-based analysis tools, several support algebraic property analyses are based on various theories. For example, the On-the-fly Modal Checker [12] explores the state space based on a requirement-driven approach. The Constraint Logic-based Attack Searcher [13] runs protocols in all possible aspects on a limited session set based on constraint logic, converting traces into constraints. The Tree Automata based on Automatic Approximations for the Analysis of Security Protocols [14] uses tree automata based on automatic approximation analysis with a rule-tree language with rewriting to approximate intruder knowledge. ProVerif, an automatic cryptographic protocol verifier [15], verifies that a protocol satisfies a set of given user attributes based on an overapproximation technology (such as the abstraction generated by a new nonce). The Tamarin Prover, a security protocol verification tool that supports both falsification and unbounded verification in the symbolic model [16], supports the Diffie-Helman (DH) method [17] and exclusive-OR (XOR) [18] theory based on protocol descriptions of multiset rewriting systems.

Scyther [19] used the strand space model to represent protocol roles and applied a pattern-based reverse search algorithm to perform bounded or unbounded attribute verification on the protocol [20]. In [21], Cremers proposed a method to approximately describe the DH operation using IKEv1 and IKEv2 protocols using an auxiliary protocol.

The strand space model [22] is a practical formal method of analysing security protocols. The theoretical basis of the strand space model was built upon the Dolev-Yao model [23] proposed by Febrag et al., which transforms the role of the state and overall process of protocol operation into a set

and directed graph to determine if attack nodes existed by deducing the set.

Automatic analysis tools using the strand space model as their theoretical bases include Athena [24], Scyther, Maude's Naval Research Laboratory Protocol Analyzer (NPA) [25], Cryptographic Protocol Shape Analyzer (CPSA) [26], and the Tamarin Prover. The strand space model is widely used for protocol analysis. Yang et al. [27] solved the representation selection problem of a strand space model, allowing protocol selection along different paths and integrated syntax and transformation rules of process algebra into Maude NPA strands. Basin and Cremers [28] extended strand space model support to an adversary model and modelled the attacker in Scyther-compromise. Dong and Niu [29] extended the anonymity analysis framework and qualitatively analysed the differences of degrees of anonymity.

In this research, our contributions are as follows.

We establish the Extended Strand Space Model (ESSM) framework with algebraic strands to represent protocol operations and use different bundles to represent different attacker behaviours. The model has scalability and protocol adaptability, and it can select attacker capabilities according to the specific protocol and communication environment while accurately modelling attacker behaviours.

We establish the semantic description of the algebraic capability of the strand space, extending the attacker's ability to obtain messages with algebraic properties. The attacker can obtain previously ignored information in protocols supporting the XOR and Abelian groups.

We establish a semantic support mechanism for special attacks against attackers. The models of algebraic attacks, key leakage attacks, and guessing attacks are carried out in specific environments.

The correctness of the added semantic logic is verified using Scyther's engineering implementation of ESSM. The applicability and correctness of the framework are illustrated by comparing the number of detection paths with the ability to detect attacks before and after addition.

The rest of this paper is organized as follows. In Section 2, we briefly review the basic definition of strand space model. The third section elaborates the establishment of the ESSM framework. Section 4 shows the performance of ESSM using real protocol analyses. Section 5 concludes the paper and discusses future work. The source code of the protocol formal model can be obtained from https://github.com/mmmxy555/ESSM.

2. Strand Space Theory

This section briefly introduces the basic concepts of strand space theory, the attacker model, and security attribute representation.

2.1. Basic Concepts. In the strand space model, the behaviours of the protocol participant and attacker are described as strands, and the set of these strands constitutes the strand space. The symbols are shown in Table 1.

Table 1: Symbols in strand space.

Symbol	Specification
Agent	Roles involved in protocol operation
K_{AB}	Long term symmetric keys of principals A and B
K_A	Long term public key of subject a
K_A^{-1}	Long term private key of principal a
$\{x\}_K$	Encrypt data x with key K
a, b	Connection of data a and B
P	Protocol intruder

We mark the set of all elements appearing in the protocol interaction as A. We refer to the elements of A as terms, which can contain one or more subterms. $t_1 \subset t$ expresses that element t_1 is a subterm of t, where t and t_1 are both terms.

Binary $\langle \sigma, a \rangle$ is a symbolic term in which $a \in A$, and $\sigma \in \{+, -\}$, which is expressed as $+a$ or $-a$. $+a$ means that the principal sends term a, and $-a$ indicates that the principal receives term a.

Definition 1 (strand). A strand is a finite sequence containing several symbolic terms. A strand s, with n symbolic terms, can be expressed as $s = \langle \langle \sigma_1, a_1 \rangle, \ldots, \langle \sigma_n, a_n \rangle \rangle$. We define the set of strands as S and set all strands in protocol P as $s_i (1 \le i \le t)$, where $s_i \in S_P$.

The strand space model was used to construct the Needham-Schroeder public key (NSPK) protocol.

(i) Initiator strand: $\text{Init}[A, B, N_A, N_B] = \langle +\{N_A, A\}_{K_B}, -\{N_A, N_B\}_{K_A}, +\{N_B\}_{K_B} \rangle$.

(ii) Responder strand: $\text{Resp}[A, B, N_A, N_B] = \langle -\{N_A, A\}_{K_B}, +\{N_A, N_B\}_{K_A}, -\{N_B\}_{K_B} \rangle$.

The strand space digraph of the protocol can be obtained by associating the collusion of each role through a causal connection. For example, in NSPK, the strand space digraph is given as Figure 1.

2.2. Protocol Attacker Description. An attacker's ability follows the attacker model defined by Dolev and Yao using discard, generate, and combine messages. In the strand space model, the attacker's ability is realized via a combination of an attacker's atomic operations, as defined in Table 2.

Bundle is a structure in the strand space, composed of some strands, connected by some binary with opposite signs but the same terms. Figure 1 can be seen as a bundle composed of two strands. The three symbol terms of each strand satisfy the same terms but opposite signs. This establishes a connection between the two strands to form a bundle.

Using initial knowledge and atomic operations, the attacker can completely control the channel, eavesdrop, tamper, or redirect messages and expand the known information of the attacker via encryption and decryption.

2.3. Representation of Security Attributes. We mainly consider that the attacker can obtain secret information protected in the protocol through a combination of attacker

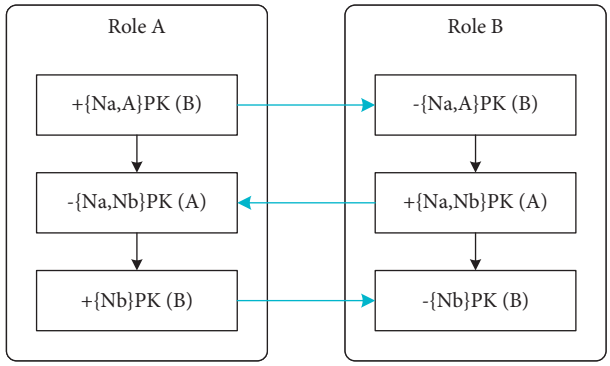

Figure 1: Strand space digraph of NSPK.

Table 2: Attacker operation of strand space.

Symbol	Specification	Strand
M	Generate text messages	$\langle +t \rangle$
F	Receive messages	$\langle -g \rangle$
T	Receive and send messages	$\langle -g, +g, +g \rangle$
C	Connect received messages	$\langle -g, -h, +gh \rangle$
S	Separating received messages	$\langle -gh, +g, +h \rangle$
K	Send key	$\langle +K \rangle$
E	Encrypted message	$\langle -K, -h, +\{h\}_K \rangle$
D	Decrypt message	$\langle -K^{-1}, -\{h\}_K, +h \rangle$

strands and initial knowledge. The confidentiality of secret information x means that there is no node, n (e.g., a normal node or an attacker node), and considers unprotected x as its term. The definition of confidentiality is as follows.

Definition 2 (secrecy). A value x is secret in a strand space Σ if, for every bundle C in Σ and for every node $n \in C$, the term $(n) \neq x$.

Additionally, authentication can also be assured. A protocol satisfies the requirement of authentication, which indicates that each subject of the protocol receives the terms that should be accepted according to the protocol expectation.

Definition 3 (authentication). A protocol guarantees a participant's (B (e.g., the responder)) agreement for certain data terms x, with participant A if, in a strand space Σ, for every bundle C, containing a responder strand using x in Σ, there exists a unique initiator strand using x in C.

A weaker noninjective agreement does not ensure uniqueness.

3. Extended Strand Space Model

3.1. ESSM Framework. We extend the semantics of the strand space model and propose the ESSM framework shown in Figure 2.

The definitions of strand and bundle inherit the definition of the strand space model. In ESSM, a strand can be divided into three types: role, algebra, and attack.

The role strand represents the sending and receiving message strands fulfilling the role of the protocol interaction. The algebra strand includes a newly defined algebraic operation strand. The attacker strand contains the original attack capability and extension capability modules.

Per the role-interaction rules defined for the protocol, the role strands describe the order of receiving and sending messages through protocol subjects. The algebra strand is a novel type added in ESSM which can be modularized and extended; it describes the conversion rules of algebraic operations in the protocol. Algebraic operations can be shared by the principal and the attacker, and the agent can use algebraic operations and basic encryption and decryption rules, E and D, to complete the internal operations of the agent. Simultaneously, the equivalent relationship of the algebraic operation can be modelled. For attackers, algebraic operations can be used to acquire more terms in the ESSM than those in the original strand space model. The basic attacker strand inherits the semantics of M, F, T, and other penetrator strands in the strand space model. In the extension module, the problems existing in the specific protocol can be combined with the model.

We use three disjoint sets to represent all the strands in protocol P. S^P_{role}, S^P_{algebra}, and S^P_{attack} refer to the set of all role, algebra, and attack strands in P, respectively. Then, the strand set satisfies $S^P_{\text{role}} \cup S^P_{\text{algebra}} \cup S^P_{\text{attack}} = S^P$.

The extended strands must satisfy the basic rules of obtaining terms, meaning that the terms obtained by an attacker must have appeared before.

Definition 4 (extended strands). Strand $\langle -\text{term}_1, -\text{term}_2, \ldots, -\text{term}_n, +\text{term}_{\text{obtain}} \rangle$ is a legal extended strand, if, for all subterm $\subset \text{term}_{\text{obtain}}$, there exists a number $i\,(1 \leq i \leq n)$ that satisfies subterm $\subset \text{term}_i$.

An extended strand space is a graph of three types of collusions connected by causal dependency. This graph is the set space of all roles, algebra operations, and attacks. We use the rules of confidentiality and authentication in the SSM to determine whether the security attributes of a protocol are satisfied.

In ESSM, the concept of algebraic rule strands that support some algebraic operations in the interaction behaviour of agents is introduced. Simultaneously, attackers can use these algebraic rule strands to carry out attacks. The algebraic operations commonly used in the XOR operation and Abelian group operations are semantically modelled such that ESSM can be used in protocols that support algebraic operations.

In the extended model, the attacker's ability is abstracted into descriptions of the attacker's behaviours using different atomic rules. This modular design enables us to define attacker models for specific protocols.

Furthermore, the algebraic and attacker rules in ESSM are extensible. Thus, ESSM can be further extended by the systematic description of atomic rules for added algebraic operations or attack capabilities.

Compared with the original strand space model, our extended framework has two advantages.

One is the formal description of algebraic properties. Traditional analysis ignores the nature of algebraic operations when there are algebraic operations in the

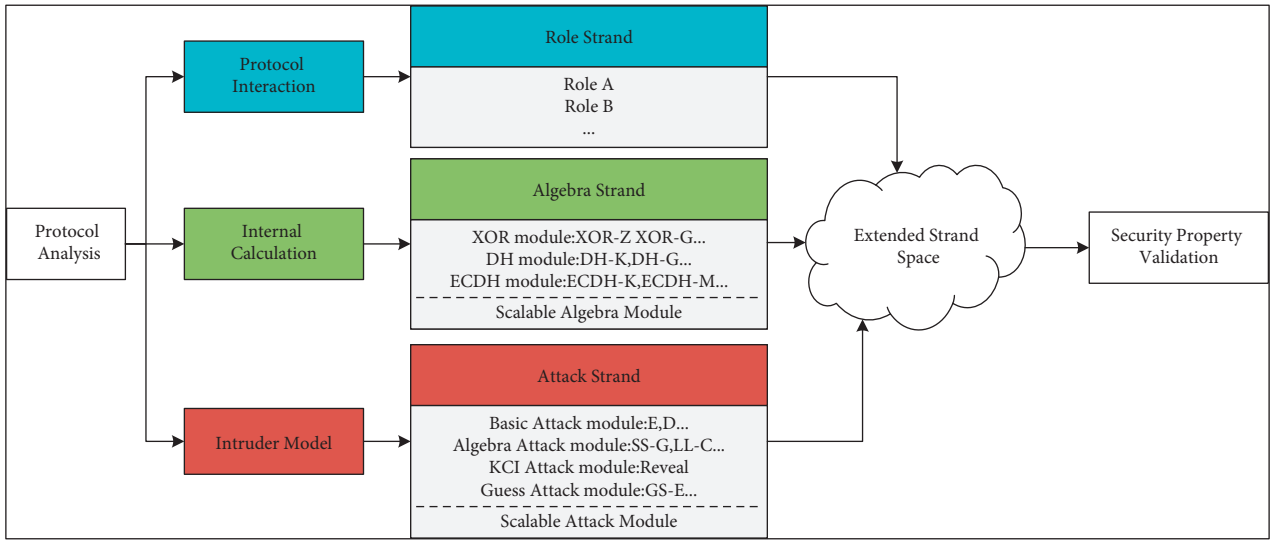

Figure 2: ESSM framework.

protocol and cannot detect attacks carried out by attackers using algebraic operations. Our extension can search for this type of attack and expand the types of protocols that can be analysed.

The second is a custom description of attack capabilities. The traditional SSM is based on the Dolev-Yao attacker model, assuming that the cryptographic primitives are unbreakable. In fact, attackers may have attacks such as KCI attacks and weak password guessing. These are not considered in traditional analysis. In our extension, we can consider different problems according to the possible problems of the protocol. The attack models can be freely combined to find the problems in the protocol.

3.2. Algebraic Attribute Addition. The basic strand space model does not support algebraic operations (e.g., XOR or Abelian groups), and attackers have no way of locating attacks related to algebraic properties. Instead, a one-way function is used to model the XOR and Abelian groups abstractly, such that the strand space model can support protocol analysis using algebraic operations, and attackers can use algebraic operation strands to detect problems with algebraic operations in protocols.

For the addition of different modules, we introduce new types of terms and functions. When the type of the term matches the type of the function parameter, the newly added strand can be applied. At the same time, the new type of strand is also compatible with the operations of traditional SSM attackers. For example, for Num type terms, the attacker can also perform operations such as generation and eavesdropping.

3.2.1. XOR Operation. The XOR operation requires the establishment of an algebraic model that satisfies the following operational relations. For $a, b, c \in \text{Num}$,

$$\begin{aligned} a \oplus 0 &= a, \\ a \oplus 0 &= a, \\ a \oplus b &= b \oplus a, \\ (a \oplus b) \oplus c &= a \oplus (b \oplus c). \end{aligned} \tag{1}$$

We use a hash function combined with a new set of decryption semantics to achieve the XOR operation. Owing to the unidirectionality of the hash function, the attacker can construct $\text{xor}(a, b)$ when a and b are known. However, terms a and b cannot be obtained through $\text{xor}(a, b)$.

Definition 5 (XOR operation). One-way function xor: $\text{Num} * \text{Num} \longrightarrow \text{Num}$; $\text{xor}(a, b)$ denotes the exclusive xor of terms a and b.

The attacker can generate $\text{xor}(a, b)$ through a and b, and if $\text{xor}(a, b)$ and b are known, the attacker can calculate $(a \oplus b) \oplus b = a$ to obtain term a, which cannot be described by the hash function. Thus, a new model of the attacker's derivation ability is needed.

We built an XOR operation module, which is shown in Table 3. For the protocol containing the XOR operation, we can add an XOR operation semantic module to model the protocol. Attackers can obtain information in an algebraic operation.

In rule XOR-Z, we construct a constant z, which represents the zero element, which is included in the initial knowledge by the subject and attacker in the protocol description. In rule XOR-G, attackers can apply XOR to construct the XOR values of two known terms. In rule XOR-S, the attacker can obtain $\text{xor}(b, a)$ using the known XOR value $\text{xor}(a, b)$. Moreover, these two terms are independent in the strand space model, and the exchange law of the XOR operation can be constructed using this rule. In rule XOR-D, an attacker can obtain the second term b, by knowing the XOR value $\text{xor}(a, b)$, and the first term a. In rule XOR-O, an attacker can obtain the term XOR from the zero element. In

Table 3: Strands of XOR operation module.

Rule name	Specification	Strand
XOR-Z	Initial knowledge	$\langle +z \rangle$
XOR-G	Generate XOR terms	$\langle -a, -b, +\text{xor}(a,b) \rangle$
XOR-S	Commutative rule of XOR	$\langle -\text{xor}(a,b), +\text{xor}(b,a) \rangle$
XOR-D	Reflexive rule of XOR	$\langle -\text{xor}(a,b), -a, +b \rangle$
XOR-Z	Obtains the term XOR with 0	$\langle -\text{xor}(a,z), +a \rangle$
XOR-C	Associative rule of XOR operation	$\langle -\text{xor}(\text{xor}(a,b),c), +\text{xor}(a,\text{xor}(b,c)) \rangle$

rule XOR-C, the attacker can apply the binding law to combine the XOR values of the three elements.

We do not have a decryption rule for the first term of the XOR model, because it can be implemented by applying the XOR-S and XOR-D rules.

Inferring that the attacker knows $\text{xor}(a,b)$ and b, term a can be obtained using these known values. The attacker first applies the XOR-S rule $\langle -\text{xor}(a,b), +\text{xor}(b,a) \rangle$, gets $\text{xor}(b,a)$, and passes the XOR-D rule $\langle -\text{xor}(b,a), -b, +a \rangle$ to obtain term a.

3.2.2. Abelian Group Operation. In security protocols, the application of an Abelian group is embodied in the key agreement algorithms, DH, and the elliptic-curve DH (ECDH). Using the multiplicative group on Z_p^* and the additive group on an elliptic curve, we analyse the properties of the Abelian group, describe the operation of the Abelian group in ESSM, and model the ability of attackers to obtain terms from the operation.

We describe the semantics of the multiplication group on Z_p^* as follows: for the primitive element g over Z_p^*,

$$\begin{aligned} (g^x)^y &\equiv g^{x*y} \pmod p, \\ g^{x*y} &\equiv g^{y*x} \pmod p, \\ g^x * g^y &\equiv g^{y+x} \pmod p, \\ g^{x+y} &\equiv g^{y+x} \pmod p. \end{aligned} \tag{2}$$

First, two one-way functions, add and mul, are defined to represent the addition and multiplication operations of the two variables.

Definition 6 (add operation). One-way function add: $Z * Z \longrightarrow Z$; $\text{add}(a,b)$ denotes the addition of terms a and b.

Definition 7 (mul operation). One-way function mul: $Z * Z \longrightarrow Z$; $\text{mul}(a,b)$ denotes multiplication of terms a and b.

Similar to the XOR operation, attackers can construct add and mul function values that support exchange to deduce the value of another element by knowing the whole function value and one element. We define these three algebraic properties as gen, swap, and decrypt, as shown in Table 4.

For the equivalence relations in the Abelian group having different forms on both sides, we use the equivalence relation $(g^x)^y \equiv g^{x*y}$. Because it is impossible to describe the equivalence relation in the strand space model, a bidirectional derivation relationship should be considered, such as $(g^x)^y \equiv g^{x*y}$ and $g^x * g^y \equiv g^{y+x}$. This equivalence relation is expressed as a bidirectional strand space model. For the subject and attacker of the protocol, we establish the semantic rules in Table 5.

We describe the DH key exchange protocol in the strand space by applying the semantics of the multiplication group on Z_p^*. Via the key exchange, the two parties can establish a shared key, $\exp(g, \text{mul}(x,y))$. The strand representations of protocol roles A and B are as follows:

(1) Initiator A's strand: $\text{Init}[A,B,x] = \langle +\exp(g,x), -\exp(g,y), +\{\text{mes}\}\ \exp(g, \text{mul}(x,y)) \rangle$.

(2) Responder B's strand: $\text{Resp}[A,B,y] = \langle -\exp(g,x), +\exp(g,y), -\{\text{mes}\}\exp(g, \text{mul}(x,y)) \rangle$.

In the role strand of this protocol, the third message between A and B is not trivial, because both parties need to obtain $\exp(g, \text{mul}(x,y))$. The message is then encrypted and decrypted. Term $\exp(g, \text{mul}(x,y))$ is obtained by adding the semantics. Considering principal A as an example, the process of obtaining the shared key is as shown in Figure 3.

Role A applies the DH-G rule to obtain the term $\exp(g,y)$. A obtains the term $\exp(\exp(g,y),x)$ by combining the initial knowledge of x. DH-L1 rules are then applied to obtain the term $\exp(g, \text{mul}(y,x))$. Then, DH-S1 rules are applied to obtain the term $\exp(g, \text{mul}(x,y))$. At this point, role A obtains the symmetric key established by both parties, and role B can obtain the term $\exp(g, \text{mul}(x,y))$ using similar methods. Roles A and B interact with the third node using the shared key.

Similarly, we establish the operation rules of the Abelian group on an elliptic curve, assuming that P is a point on the elliptic curve E, and $x, y \in Z$. P has the following properties:

$$\begin{aligned} x \times (y \times P) &= (x * y) \times P, \\ (x * y) \times P &= (y * x) \times P, \\ x \times P + y \times P &= (x + y) \times P, \\ (x + y) \times P &= (y + x) \times P. \end{aligned} \tag{3}$$

We define addition and multiplication on elliptic curves as ecadd and ecmul, which are distinguishable from add and mul which are defined above.

Definition 8 (ecadd, ecmul operation). One-way function ecadd: Point × Point ⟶ Point; ecmul: $Z \times \text{Point} \longrightarrow$ Point; $\text{ecadd}(a,b)$ denotes the addition operation of terms a and b of type point, and the term type obtained is point;

Table 4: Strands of ADD and MUL operation module.

Rule name	Specification	Strand
ADD-G	Generative addition terms	$\langle -a, -b, +\text{add}(a,b)\rangle$
ADD-S	Commutative rule of addition	$\langle -\text{add}(a,b), +\text{add}(b,a)\rangle$
ADD-D	Inverse rule of addition	$\langle -\text{add}(a,b), -a, +b\rangle$
MUL-G	Generative multiplication terms	$\langle -a, -b, +\text{mul}(a,b)\rangle$
MUL-S	Commutative rule of multiplication	$\langle -\text{mul}(a,b), +\text{mul}(b,a)\rangle$
MUL-D	Inverse rule of multiplication	$\langle -\text{mul}(a,b), -a, +b\rangle$

Table 5: Strands of DH operation module.

Rule name	Specification	Strand
DH-K	Initial knowledge	$\langle +g\rangle$
DH-G	Perform power module operation	$\langle -a, -b, +\exp(a,b)\rangle$
DH-S1	Commutative rule of multiplication	$\langle -\exp(g, \text{mul}(a,b)), +\exp(g, \text{mul}(b,a))\rangle$
DH-S2	Commutative rule of addition	$\langle -\exp(g, \text{add}(a,b)), +\exp(g, \text{add}(b,a))\rangle$
DH-L1	Deduction rule L1	$\langle -\exp(\exp(g,a),b), +\exp(g, \text{mul}(a,b))\rangle$
DH-L2	Deduction rule L2	$\langle -\exp(g, \text{mul}(a,b)), +\exp(\exp(g,a),b)\rangle$
DH-R1	Deduction rule R1	$\langle -\exp(g, \text{mul}(a,b)), +\exp(\exp(g,a),b)\rangle$
DH-R2	Deduction rule R2	$\langle -\exp(g, \text{add}(a,b)), +\text{mul}(\exp(g,a), \exp(g,b))\rangle$

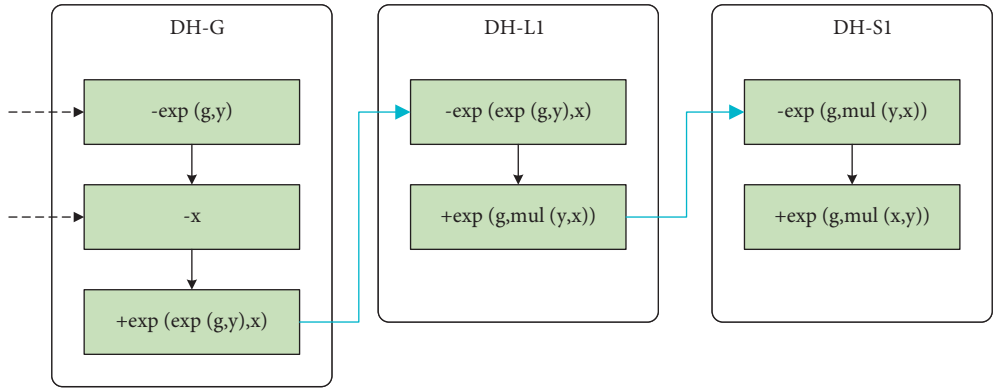

Figure 3: The path for legal agent to obtain shared key.

ecmul (a,b) represents multiplication operation of the term of type Z and the term of type point, and the term type obtained is point.

Similarly, the operation rules of the elliptic curve are established in Table 6.

3.3. Attacker Capability. In this section, we extend the attacker attack model using modularization. Based on the classic Dolev-Yao model, in the first section, we model a variety of attacks based on algebraic properties, including small group attacks, Lim-Lee attacks, and others that need to be combined with group properties. The second section introduces the extension of the key-compromise impersonation (KCI) attack, which can describe the situation of specific information exposure. The third section considers the influence of guessing attacks on security protocols and formalizes the attack.

3.3.1. Attack Based on Algebraic Form. In this section, we describe the algebraic attacks which have been already shown to exist, including subgroup attacks and Lim-Lee attacks. We reveal that the attacker can destroy algebraic properties in a specific environment to obtain secret information.

(1) Small-Group Attack. The small-group attack was first proposed by van Oorschot and Wiener [30]. This type of attack takes advantage of the structural characteristics of a group to replace the key negotiated by both sides of the communication. The negotiated key can be obtained without affecting the normal communication between the two sides.

In the implementation, if the Abelian group used in the protocol is Z_p^*, order $p-1$ is a composite number. If the order of group G used in the protocol is a composite number, $n = r * \omega$, assuming that r is a small factor of n, and G has subgroups $\langle g^t\rangle^*$, meaning that g^t is a multiplicative group whose generator order is r. There are only r elements in this group.

If the shared secret key negotiated by both parties is in group $\langle g^t\rangle^*$, the attacker can guess the real key exhaustively when the two parties communicate with each other using key encryption.

Table 6: Strands of ECDH operation module.

Rule name	Specification	Strand
ECDH-K	Initial knowledge	$\langle +P \rangle$
ECDH-G	Perform multiplication	$\langle -a, -P, +\text{ecmul}(a, P) \rangle$
ECDH-S1	Addition exchange rule	$\langle -\text{ecmul}(\text{add}(a, b), P), +\text{ecmul}(\text{add}(b, a), P) \rangle$
ECDH-S2	Multiplication exchange rules	$\langle -\text{ecmul}(\text{mul}(a, b), P), +\text{ecmul}(\text{mul}(b, a), P) \rangle$
ECDH-L1	Deduction rule L1	$\langle -\text{ecmul}(a, \text{ecmul}(b, P)), +\text{ecmul}(\text{mul}(a, b), P) \rangle$
ECDH-L2	Deduction rule L2	$\langle -\text{ecadd}(\text{ecmul}(a, P), \text{ecmul}(b, P)), +\text{ecmul}(\text{add}(a, b), P) \rangle$
ECDH-R1	Deduction rule R1	$\langle -\text{ecmul}(\text{mul}(a, b), P), +\text{ecmul}(a, \text{ecmul}(b, P)) \rangle$
ECDH-R2	Deduction rule R2	$\langle -\text{ecmul}(\text{add}(a, b), P), +\text{ecadd}(\text{ecmul}(a, P), \text{ecmul}(b, P)) \rangle$

Considering the simple DH as an example, the attack process is as follows:

(1) Role A initiates DH key exchange with role B, generates a random number x, and calculates the public key, g^x.

(2) The attacker intercepts g^x and calculates $g^{\omega x}$ sent to B.

(3) Role B receives message $g^{\omega x}$. Subsequently, random number y is generated and the public key g^y is calculated and sent to C. The negotiated key is calculated as $Z_{BA} = g^{(\omega x)y} = g^{\omega xy}$.

(4) The attacker intercepts g^y and calculates $g^{\omega y}$ sent to A.

(5) Role A receives a message $g^{\omega y}$. Then, the key is calculated as $Z_{AB} = (g^{\omega y})^x = g^{\omega xy}$.

(6) Roles A and B use Z_{AB} as a session key for message passing, and the attacker eavesdrops the encrypted message and guesses to verify the session key, $g^{\omega xy}$.

Assuming that the order of group G used in the protocol is a composite number, the attacker can decompose it to obtain ω. Then, for all $g^{x\omega}$ for symmetric key messages, attackers can obtain x and key g^x by the exhaustive computation of $g^{\omega x}$. We ignore the details of the exhaustive computation and assume that the attacker can decompose a large integer, n. We model the derivation relationship of the attacker in Table 7. For the operation relationship of DH, we extend the derivation in the previous section.

Rule SS-G means that the attacker will use the term on group $\exp(g, a)$ through the operation of the ω-power module on the subgroups to obtain terms $\exp(g, \text{mul}(a, \omega))$. The SS-V rule indicates that the attacker obtains the elements, $\exp(g, \text{mul}(a, \omega))$, on the subgroup via exhaustive verification using the elements in the subgroup as the term for key encryption. The type of term a is not limited. For example, the key negotiated by both sides of DH protocol under the attack of small groups can be $\exp(g, \text{mul}(\text{mul}(x, y), \omega))$. In this case, a in the formula means $\text{mul}(x, y)$.

(2) Lim-Lee Attack. Owing to the discovery of small group attacks, a preventive measure uses the prime, q, subgroup of Z_p^*. However, Lim and Lee found an attack method having prime order [31] against the group. Thus, the attacker can obtain the private key of the responder role by actively participating in the operation of the protocol.

Taking simple DH as an example, the attack process is as follows:

(1) Attacker C initiates a DH key exchange with role B to generate random numbers $x \in Z_q^*$ and calculate the public key, g^x. Simultaneously, β is generated. The order of β is r and it satisfies $r|(p-1)/q$. C sends $\beta * g^x$ to role B.

(2) Role B receives the message, $\beta * g^x$. Subsequently, random number, y, is generated, and the public key, g^y, is calculated and sent to C. The negotiated key is $Z_{BC} = (\beta * g^x)^y = \beta^y * g^{xy}$.

(3) Attacker C receives g^y and calculates $Z_{CB} = (g^y)^x = g^{x*y} = Z_{BC}/\beta^y$, because β^y is the only r available for the attacker to use to obtain the correct partial information, $y(\text{mod}r)$, by verification.

(4) By trying β corresponding to different r, the attacker can obtain equations with different moduli, and the complete information of y can be obtained using the Chinese remainder theorem.

Assuming that the group used in the protocol is a prime group of order q and that the attacker can participate in and initiate the protocol, we define a strand space model for the attacker to execute the Lim-Lee attack in Table 8.

Rule LL-G converts the elements, $\exp(g, a)$, of a group beyond the group to obtain $\text{mul}(\beta, \exp(g, a))$, where the order of β is r and it satisfies $r|(p-1)/q$. Rule LL-V obtains the information of $y(\text{mod}\, r)$ by guessing its verification, ignoring the specific guessing process, and it assumes that the attacker's guessing ability can calculate the data of scale r. Rule LL-C uses the Chinese remainder theorem to recover the complete information of y, thereby creating a sample. Moreover, the attacker must use different r_i of β_i, conduct intrusion behaviours, and obtain one term, $y(\text{mod}\, r_i)$, at a time. Through different r_i, it combines complete information on y. We abstract this process, and, to preserve the principle of attack, we express this process as the term, y, calculated by $y(\text{mod}\, r_i)$. The attacker only needs to execute one intrusion to obtain term y.

3.3.2. KCI Attack. It is possible for an attacker to break through a device to obtain its long-term private key, or in a protocol using a smart card, to obtain the smart card of a legitimate subject leading to a smart-card loss attack. We define this behaviour semantically and describe it as a KCI attack.

Table 7: Strands of small group attack module.

Rule name	Specification	Strand
SS-G	Generating elements on subgroups	$\langle -\exp(g,a), +\exp(g,\mathrm{mul}(a,\omega))\rangle$
SS-V	Guess the elements on a subgroup	$\langle -\{\mathrm{mes}\}\exp(g,\mathrm{mul}(a,\omega)), +\exp(g,\mathrm{mul}(a,\omega))\rangle$

Table 8: Strands of Lim-Lee attack module.

Rule name	Specification	Strand
LL-G	Generating elements outside the group	$\langle -\exp(g,a), +\mathrm{mul}(\beta,\ \exp(g,a))\rangle$
LL-V	Guess the elements of validation	$\langle \{\mathrm{mes}\}\exp(\mathrm{mul}(\beta,\ \exp(g,a)), y), +\mathrm{mod}(y)\rangle$
LL-C	Computing term y using CRT	$\langle -\mathrm{mod}(y), +y\rangle$

An attacker can obtain the long-term private key, session key, or some state in communication by corroding the agent or via cryptanalysis. We model this ability as a message, mes, which the attacker steals from the role strand. Hence, the derivation relation of some information in the message that cannot be obtained directly can then be obtained. The term, key, can be symmetric or asymmetric, encrypted or hashed, or more complex. The attackable information must be included in the role strands. Thus, this method has a certain applicability that further indicates the situation of secret information disclosure (e.g., role long-term private key, session key, and smart card). The corresponding disclosure rules must be generated alongside specific protocols. Here, only the framework of the attacker rules is given in Table 9.

Note that these rules must be implemented in combination with specific protocol role strands. For details, refer to the KCI attack and impersonal attack in the MTI protocol in the next section, as well as the analysis of the three-factor authentication protocol. For specific protocols, the terms of disclosure can be specified.

3.3.3. Guessing Attack. Guessing attacks include two parts. First, an attacker intercepts a message related to the value to be guessed. Then, the attacker matches the correct guess value by traversing the dictionary. The default value to be guessed is a password. For the first part, the attacker must have a detection method that can verify the conjecture.

(i) Password as encryption key

If the attacker can obtain a message pair similar to term and {term}pw, the attacker can generate {term}pw′ by constructing pw′ and encrypting semantic E to verify that the generated key pw′ is correct by comparing whether {term}pw′ and {term}pw are consistent.

(ii) Password contained in a hash

If the attacker can obtain a $\mathrm{hash}(\mathrm{pw}, \mathrm{term}_1, \ldots, \mathrm{term}_n)$ and knows $\mathrm{term}_i\ (1 \le i \le n)$, the attacker can guess the password, pw′, and construct a hash, $\mathrm{hash}(\mathrm{pw}', \mathrm{term}_1, \ldots, \mathrm{term}_n)$. The original hash value is compared to verify the guess.

In the second part, the success rate of guess attack depends on the complexity of the password set by the agent and the size of the dictionary used. Theoretically, if the password is in the dictionary, it can be successfully cracked. In the theoretical description of guessing attacks, the attacker has enough elements in the dictionary to carry out a guessing attack on any message that meets the requirements.

Table 9: Strands of KCI attack module.

Rule name	Specification	Strand
Reveal	Get some key from encryption	$\langle -\{\mathrm{mes}\}\mathrm{key}, +\mathrm{key}\rangle$
Reveal	Get some terms from encryption	$\langle -\{\mathrm{mes}\}\mathrm{key}, +\mathrm{mes}\rangle$
Reveal	Get some keys from hash	$\langle -\mathrm{hash}(\mathrm{key}), +\mathrm{key}\rangle$

However, the situation in the real protocol may be more complex. For example, the password is used as the key after hashing or multiple hashings, which can be regarded as the multiple effects of the basic situation. In short, the attacker can crack the weak password after obtaining a message that meets the guessing condition. The formal description of this ability is shown in Table 10.

In accordance with the description of a KCI attack, the definition of the attacker strand of a guessing attack should be combined with a specific protocol. In rules GS-E and GS-H, only the description method of the guessing attack is described. It is thus necessary to combine the strand of the subject to customize the ability of the attacker to carry out a guessing attack.

4. Implementation and Experimental Results

We implemented support for ESSM and applied Scyther to test a set of protocols that use algebraic operations and an extended attack capability. In this section, we describe our implementation and experimental results.

4.1. Implementation. We implemented the ESSM model using Scyther (version 1.1.3). Our implementation used auxiliary rules as additional input, combined with the definition of the protocol body to form a Security Protocol Description Language (SPDL) file as the input for Scyther model checking. We expanded the original protocol in algebraic operation and attack ability including (1) the running rules of the protocol body, (2) added algebraic operation rules, (3) added attacker rules, and (4) defining the security attributes of the check. Additionally, options can be

Table 10: Strands of guessing attack module.

Rule name	Specification	Strand
GS-E	Guess password from encryption	$\langle -\{a\}\text{pw}, -a, +\text{pw}\rangle$
GS-H	Guess password from hash	$\langle -\text{hash}(a, \text{pw}), -a, +\text{pw}\rangle$

added to Scyther, such as outputting proof procedures and limiting the number of computing processes.

Our implementation followed the ESSM construction described in Section 3 by precisely formalizing the algebraic properties and the attacker's special attack ability.

Considering the XOR attribute as an example, the following describes the process of converting the atomic rule of the XOR operation into the auxiliary rule input of Scyther.

We used an auxiliary protocol to represent an algebraic operation or attack capability module. Under each auxiliary protocol, each role represents an atomic rule. Attackers call a combination of several rules in different auxiliary protocols to implement their attack behaviours. Specific auxiliary protocol input files should be established according to specific protocol interactions for some special attacks, such as key disclosure and guessing attacks.

Moreover, without these auxiliary protocols, the interaction of protocol entities modelled by the original Scyther can work normally. However, Scyther cannot find the problems in the protocol. Compared with the original protocol modelling, using ESSM to model and analyse the protocol can find potential algebraic logic problems and special attack paths within the protocol.

4.2. Sample Protocol. We used extended strand space semantics to describe several protocols (e.g., three-factor authentication). We found known attack paths and revealed new ones.

Taking the three-factor authentication protocol proposed by Zhang et al. [32] as an example, two attack paths were successfully analysed using the extended algebraic property semantics and the attacker's ability. One was found by Mao et al. [33], and the other is the undiscovered attack path. The discovery of the two attack paths combined the XOR, key-compromise attack, and guessing attack ability rules added to the ESSM.

4.2.1. Protocol Description. During the registration stage, the user sends the protected identity information to the server, and the server stores it and issues a smart card for authentication. It should be noted that the communication in the registration phase is based on the secure channel, and the attacker cannot obtain any information in the registration phase.

(1) User U selects identity ID and password PW and inputs biometric B to the terminal. The terminal calculates $C_1 = h(\text{ID}, \text{PW}, h_{\text{Bio}}(B))$ and generates random number r_1 for calculating $C_2 = B \oplus r_1$. User U sends a registration request message $\{C_1, C_2\}$ to server S.

(2) After receiving the registration request message from user U, server S uses the server's private key, s, to calculate $M = h(h_{\text{Bio}}(C_2) \| s)$, generates random number r_2, and calculates $W = h(h_{\text{Bio}}(C_2 \oplus r_2))$, $X = h(\text{ID}_{SC} \| C_1 \| M) \oplus r_2$, and $Y = M \oplus C_1$. The server stores $\{C_2, W_0, W\}$ in the database and initializes W_0 to null. The server writes to the smart card $\{\text{ID}_{SC}, h(\cdot), h_{\text{Bio}}(\cdot), X, Y\}$ and gives it to user U.

(3) $Z = r_1 \oplus h_{\text{Bio}}(B)$ is calculated after receiving the smart card. Z is written to it for completing registration.

The login authentication phase is described as follows:

(1) User U inputs accounts for ID′, password PW′, and biometric B' and inserts the smart card at the same time.

(2) User U generates a random number r_3 and calculates $C_1^* = h(\text{ID}' \| \text{PW}' \| h_{\text{Bio}}(B'))$, $M^* = Y \oplus h(C_1^*)$, $r_2^* = X \oplus h(\text{ID}_{SC} \| C_1^* \| M^*)$, and $r_1^* = Z \oplus h_{\text{Bio}}(B)$.

(3) $C_3 = h_{\text{Bio}}(B' \oplus r_1^* \oplus r_2^*)$, $C_4 = B' \oplus r_1^* \oplus h(M^* \| r_3)$, and $C_5 = r_3 \oplus h_{\text{Bio}}(B' \oplus r_1^*)$ are calculated to send a login request $\{C_3, C_4, C_5\}$ to server S.

(4) The server performs dynamic verification by matching C_3 and the data in the database. For more details, please refer to the original article [32].

(5) Server S generates a random number r_4, calculates $M' = h(h_{\text{Bio}}(C_2) \| s)$, $r_3^* = C_5 \oplus h_{\text{Bio}}(C_2)$, and $B' \oplus r_1^* = C_4 \oplus h(M' \| r_3^*)$, and checks $B \oplus r_1^*$ and C_2. If the validation passes, the server computes $C_6 = r_4 \oplus h(B \oplus r_1^*)$ and $C_7 = h((B \oplus r_1^*) \| r_3^* \| r_4^*)$. Both $\{C_6, C_7\}$ are sent to user U.

(6) User U receives $\{C_6, C_7\}$ and calculates $r_4^* = C_6 \oplus h(B \oplus r_1^*)$, verifying $C_7 = ?h((B \oplus r_1^*) \| r_3^* \| r_4^*)$. After verification, the user calculates the session key, $\text{SK} = h(M^* \| r_3 \| r_4^*)$, and $C_8 = h(h_{\text{Bio}}(B \oplus r_1^* \oplus r_4^*) \oplus r_4^*)$, and sends the authentication message, C_8, to the server.

(7) Server receives C_8 after validation and accepts session key SK after successful verification; then it sends a key confirmation message, $C_9 = h(\text{SK} \| r_4)$, to the user.

(8) Server receives C_9 after validation. After successful verification, both parties establish a common session key, SK.

4.2.2. Strand Space Analysis of Protocol. We used the XOR operation in this protocol. Mao's analysis of the protocol included smart card and guess attacks. We added the corresponding key-compromise and guess attack modules. The behaviours of the main body of the protocol are described as shown in Figure 4.

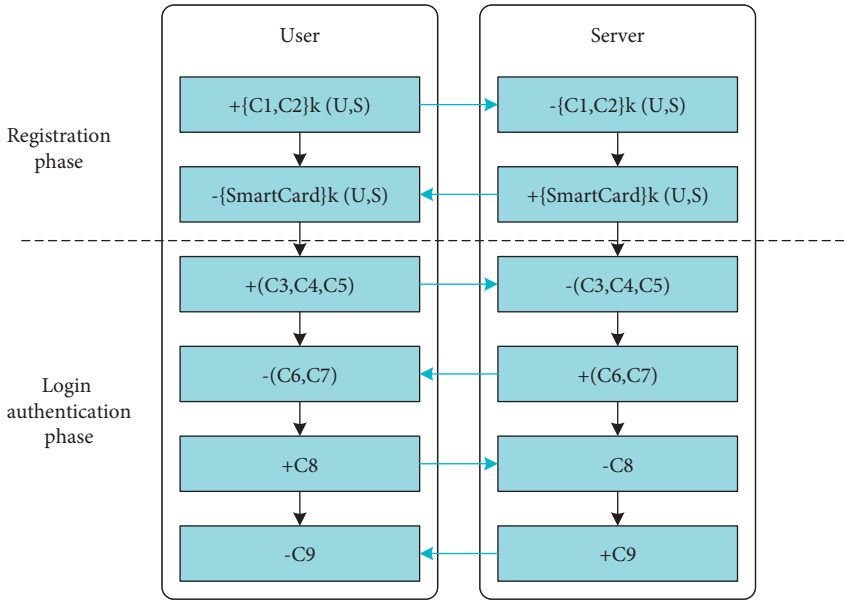

Figure 4: Role strands of 3FA.

The terms are defined as follows: $C1 = h(\mathrm{ID}u, \mathrm{PW}, H\mathrm{bio}(B))$, $C2 = \mathrm{xor}(B, r1)$, $M = h(\mathrm{Hbio}(c2), s)$, $X = \mathrm{xor}(h(\mathrm{IDs}, c1, M), r2)$, $Y = \mathrm{xor}(M, c1)$, $Z = \mathrm{xor}(r1, \mathrm{Hbio}(B))$, SmartCard = $(\mathrm{ID}s, X, Y, Z)$, $C3 = \mathrm{Hbio}(\mathrm{xor}(B, r1, r2))$, $C4 = \mathrm{xor}(B, r1, h(M, r3))$, $C5 = \mathrm{xor}(r3, H\mathrm{bio}(\mathrm{xor}(B, r1)))$, $C6 = \mathrm{xor}(r4, h(\mathrm{xor}(B, r1)))$, $C7 = h(\mathrm{xor}(B, r1), r3, r4)$, SK $= h(M, r3, r4)$, $C8 = h(\mathrm{xor}(H\mathrm{bio}(\mathrm{xor}(B, r1, r4)), r4))$, and $C9 = h(SK, r4)$.

In [33], the attack on user ID and PW required the attacker to obtain the user's smart card, know the user's biometrics, and could guess attacks. Using the framework of key-compromise and guessing attacks defined in the previous section, combined with the principal behaviours of the protocol, we modelled the attacker's ability.

The attacker knows the user's biometrics and smart cards. For the first time, biometrics appear in the first node. The smart card is divided into two parts. Z can be obtained via the XOR of $r1$ and $H\mathrm{bio}(B)$ in the first node, and the rest is sent to the user at the second node on the server. Therefore, we describe an attacker's key-compromise attack on smart cards and biometrics as Reveal:

(i) Reveal: $\langle -\{C1, C2\}k(\mathrm{User}, \mathrm{Server}), -\{\mathrm{SmartCard}\}k(\mathrm{User}, \mathrm{Server}), +(\mathrm{SmartCard}, B)\rangle$.

Modelling guessing attacks requires consideration of the terms, including IDu and password. Term $C1 = h(\mathrm{ID}u, \mathrm{PW}, H\mathrm{bio}(B))$ contains IDu and password. Furthermore, we need to obtain term B to estimate $C1$. Because of term $Y = \mathrm{xor}(M, C1)$, the known terms $C1$ and Y can obtain M by using rule XOR-D, thus conjecturing M. Two effective guessing chains can be obtained by constantly exploring the possible guessing paths. For example, in the first guess chain, we provided a set of guess values for IDu and PW, $\mathrm{ID}u_{\mathrm{Guess}}$ and $\mathrm{PW}_{\mathrm{Guess}}$, respectively. Then, combined with term B, $C1$ and the conjecture can be obtained using $C1_{\mathrm{Guess}} = h(\mathrm{ID}u_{\mathrm{Guess}}, \mathrm{PW}_{\mathrm{Guess}}, H\mathrm{bio}(B))$. Unless we obtain the conjecture value of $C4$, we cannot determine if the conjecture value is successful by comparison.

Through two guessing chains shown in Figure 5, the rules of attacker guessing attack are stated as follows:

(1) GS-1: $\langle -(C3, B, Y, \mathrm{ID}sc, X, r1), +(\mathrm{PW}, \mathrm{ID}u)\rangle$.

(2) GS-2: $\langle -(C4, B, Y, r1, r3), +(\mathrm{PW}, \mathrm{ID}u)\rangle$.

The smartcard and B obtained by combining guessing rules and key disclosures can guess the user's ID and password. The process is as follows:

Path 1 (Figure 6)

(1) The attacker obtains smartcard and B using the Reveal rule.
(2) XOR-D rule $\langle -\mathrm{xor}(r1, H\mathrm{bio}(B)), -H\mathrm{bio}(B), +r1\rangle$ is applied to obtain term $r1$.
(3) F rule $\langle -C3\rangle$ is applied to obtain term $C3$.
(4) According to the GS-1 rule, using IDsc, X, Y, B is obtained by Reveal, $r1$ is obtained by XOR operation, and $C3$ is eavesdropped using the normal protocol process. A guessing attack is carried out to obtain the terms, ID and PW, of the attacker.

Path 2 (Figure 7)

(1) The attacker obtains smartcard and B using the Reveal rule.
(2) XOR-D rule $\langle -\mathrm{xor}(r1, H\mathrm{bio}(B)), -H\mathrm{bio}(B), +r1\rangle$ is applied to obtain term $r1$.
(3) $\mathrm{Hbio}(\mathrm{xor}(B, r1))$ is constructed by B and $r1$.
(4) XOR-D rule $\langle -\mathrm{xor}(r3, H\mathrm{bio}(\mathrm{xor}(B, r1))), -H\mathrm{bio}(\mathrm{xor}(B, r1)), +r3\rangle$ is applied to obtain term $r3$.
(5) The F receive rule $\langle -C4\rangle$ is applied to obtain term $C4$.

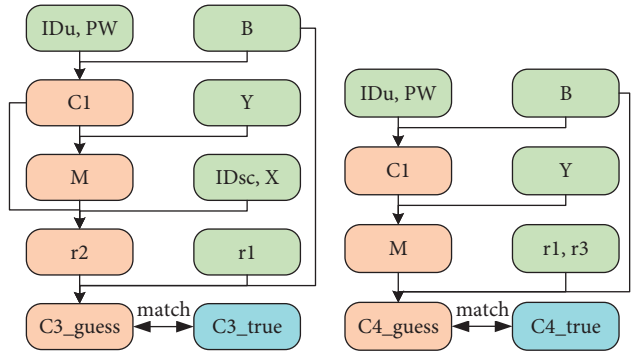

Figure 5: Two guessing chains of Zhang's protocol.

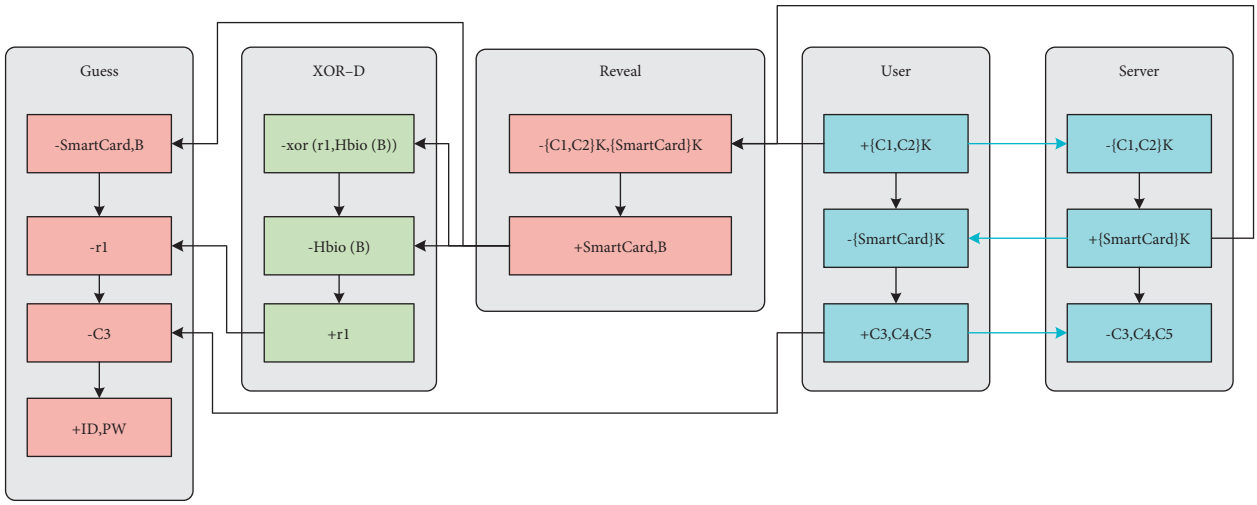

Figure 6: Attack path 1 of Zhang's protocol.

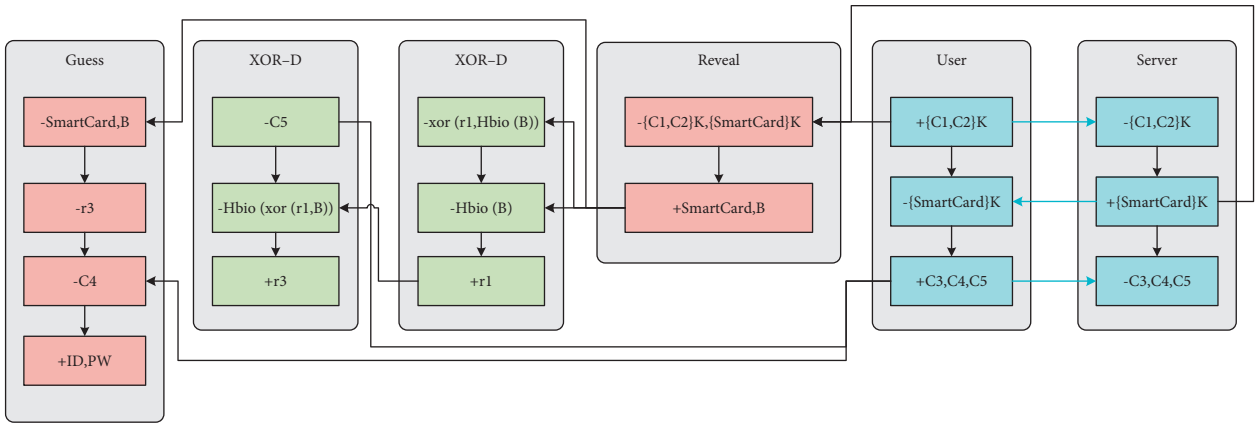

Figure 7: Attack path 2 of Zhang's protocol.

(6) According to the GS-2 rule, using Y and B obtained by Reveal, $r1$ and $r3$ obtained by the XOR operation and $C4$ from normal protocol flow are used to carry out a guessing attack to obtain the legal user's term ID and PW.

The first path was first discovered by Mao, and the second attack path was discovered by our addition of semantics to ESSM for the first time.

4.3. Experiment. We make a formal analysis of six groups of protocols, including TMN protocol [34], MTI-C (1), MTI-A (0), and MTI-C (0) in MTI protocol family [35], WPA-PSK protocol [36] in 802.11i standard, and three-factor authentication protocol proposed by Zhang.

We applied our method to a group of protocols using algebraic logic or special attack ideas. The results obtained by running our implementation on Scyther v. 1.1.3 are presented in Table 11, which lists the analysis results using the

Table 11: Experiment result of several protocols.

Protocol	Original model		ESSM model		
	Search queries	Detected attacks	Semantics used	Search queries	Detected attacks
TMN	2658	Safe	XOR	191948	Algebraic attack
MTI-C (1)	263550	Safe	DH SS	493270	Small subgroup attack
MTI-A (0)	1649	Safe	DH KCI	331160	Impersonal attack
MTI-C (0)	322	Safe	DH KCI	2585	KCI attack
WPA-PSK	269	Safe	Guess	55446	Guessing attack
Zhang's	5684	Safe	XOR KCI Guess	6349113	Combination attack

original Scyther and using ESSM modelling, including the declaration of security attributes and the number of search states.

The code restores the process of protocol interaction and abstracts the storage verification process of the server. We then declare the confidentiality of ID and PW. By adding auxiliary protocols (e.g., Smartcard Lost, XOR operation, and Offline Password Guess), two paths not meeting the confidentiality requirements can be automatically searched.

Through the experimental results, we can find that the search path of the model search after adding ESSM semantics is richer, more attacks can be found, and the protocol environment can be restored more realistically. The increase of search path shows two facts.

(1) The semantic extension of ESSM is real and effective and has certain effect on many types of protocols.

(2) The contrast of the experimental results before and after the expansion is too large, which leads to the state explosion problem to a certain extent. The current model detection technology still has no effective solution to the state explosion problem, especially for algebraic operations.

The semantic extension of ESSM is real and effective and has certain effect on many protocols.

5. Conclusion

In this paper, an ESSM framework was proposed, because it has a more complete semantic description than does the original strand space model, including the internal operation of protocol subject behaviours, the support of algebraic operation, and its modelling of the DY attacker ability. The proposed ESSM supports the transformation of algebraic operation rules at the symbol level and the expansion of a special attack capability. We added XOR- and Abelian-group operations to the algebraic operation module and added the description semantics of an algebraic attack, a KCI attack, and a guessing attack in special situations to the attacker module. The framework presented good expansibility. Furthermore, only ability rules needed to be added to the corresponding modules. Then, the corresponding protocols could be modelled and analysed in the strand space. We used ESSM to model and analyse different types of protocols that use algebraic rules and have special attack problems. We found no security or authentication problems in the strand space model, but we did encounter issues in the ESSM model. Simultaneously, we used Scyther to extend the modelling of ESSM and analysed several protocols automatically. The analysis showed that Scyther v. 1.1.3 found all problems in the protocol after modelling with ESSM. Moreover, we found a new guessing tool path using Mao's three-factor authentication protocol.

We observed that, with the extension of automation tools, the number of search paths for protocols increased. On one hand, it reflects that our model more comprehensively considers the problems in the protocol and has more search paths. On the other hand, it exposes the state explosion problem of model-checking methods, especially when dealing with algebraic operations that lead to many useless queries in the state space search. This problem will be solved in future studies.

Data Availability

The data used to support the findings of this study can be found at https://github.com/mmmxy555/ESSM.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] M. Barbosa, G. Barthe, K. Bhargavan et al., "SoK: computer-aided cryptography," in *Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP)*, pp. 777–795, IEEE, San Francisco, CA, USA, May 2021.

[2] K. Hofer-Schmitz and B. Stojanović, "Towards formal verification of IoT protocols: a Review," *Computer Networks*, vol. 174, no. 19, Article ID 10723, 2020.

[3] D. Basin, J. Dreier, L. Hirschi, S. Radomirovic, and R. Sasse, "A formal analysis of 5G authentication," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 1383–1396, ACM, New York, NY, United States, Oct 2018.

[4] C. Cremers and M. Dehnel-Wild, "Component-based formal analysis of 5G-AKA: channel assumptions and session confusion," in *Proceedings of the Network and Distributed Systems (NDSS) Symposium*, Germany, Jan 2019.

[5] C. Cremers, M. Horvat, J. Hoyland, S. Scott, and T. V. D. Merwe, "A comprehensive symbolic analysis of TLS 1.3," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 1773–1788, ACM, New York, NY, United States, Oct 2017.

[6] B. Blanchet, "Composition theorems for CryptoVerif and application to TLS 1.3," in *Proceedings of the 2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pp. 16–30, IEEE, Oxford, UK, July 2018.

[7] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila, "A formal security analysis of the signal messaging protocol," *Journal of Cryptology*, vol. 33, no. 4, pp. 1914–1983, 2020.

[8] T. Klenze, C. Sprenger, and D. Basin, "Formal verification of secure forwarding protocols," in *Proceedings of the 2021 IEEE 34th Computer Security Foundations Symposium (CSF)*, pp. 1–16, IEEE, Dubrovnik, Croatia, June 2021.

[9] C. Jacomme and S. Kremer, "An extensive formal analysis of multi-factor authentication protocols," *ACM Transactions on Privacy and Security*, vol. 24, no. 2, pp. 1–34, 2021.

[10] V. Cheval, S. Kremer, and I. Rakotonirina, "DEEPSEC: deciding equivalence properties in security protocols theory and practice," in *Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP)*, pp. 529–546, IEEE, San Francisco, CA, USA, 2018.

[11] J. Dreier, C. Duménil, S. Kremer, and R. Sasse, "Beyond subterm-convergent equational theories in automated verification of stateful protocols," *Lecture Notes in Computer Science*, vol. 10204, pp. 117–140, 2017.

[12] D. Basin, S. Mödersheim, and L. Viganò, "OFMC: a symbolic model checker for security protocols," *International Journal of Information Security*, vol. 4, no. 3, pp. 181–208, 2005.

[13] M. Turuani, "The CL-atse protocol analyser," in *Proceedings of the International Conference on Rewriting Techniques and Applications*, pp. 277–286, Springer, Nancy, France, Aug 2006.

[14] A. Armando, D. Basin, Y. Boichut et al., "The AVISPA tool for the automated validation of internet security protocols and applications," *Computer Aided Verification*, vol. 3576, pp. 281–285, 2005.

[15] B. Blanchet, "Efficient cryptographic protocol verifier based on prolog rules," in *Proceedings of the 14th IEEE Computer Security Foundations Workshop (CSFW)*, pp. 82–96, IEEE, Cape Breton, NS, Canada, June 2001.

[16] B. Schmidt, S. Meier, and C. Cremers, "Automated analysis of diffie-hellman protocols and advanced security properties," in *Proceedings of the 2012 IEEE 25th Computer Security Foundations Symposium (CSF)*, pp. 78–94, IEEE, Cambridge, MA, USA, June 2012.

[17] C. Cremers and D. Jackson, "Prime, order please! Revisiting small subgroup and invalid curve attacks on protocols using Diffie-Hellman," in *Proceedings of the 2019 IEEE 32nd Computer Security Foundations Symposium (CSF)*, pp. 78–93, IEEE, Hoboken, NJ, USA, June 2019.

[18] J. Dreier, L. Hirschi, S. Radomirovic, and R. Sasse, "Automated unbounded verification of stateful cryptographic protocols with exclusive OR," in *Proceedings of the 2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pp. 359–373, IEEE, Oxford, UK, July 2018.

[19] C. Cremers, "Scyther Tool: verification, falsification, and analysis of security protocols," in," *International Conference on Computer Aided Verification (CAV)*, pp. 414–418, 2008.

[20] C. Cremers, "Unbounded verification, falsification, and characterization of security protocols by pattern refinement," in *Proceedings of the 15th ACM conference on Computer and communications security (CCS)*, pp. 119–128, ACM, New York, NY, United States, 2008.

[21] C. Cremers, "Key exchange in IPSec revisited: formal analysis of IKEv1 and IKEv2," *Computer Security - ESORICS 2011*, vol. 6879, pp. 315–334, 2011.

[22] F. T. J. Fabrega, J. C. Herzog, and J. D. Guttman, "Strand space: why is security protocol corrects," in *Proceedings of the 1998 IEEE Symposium on Security and Privacy (SP)*, pp. 160–171, IEEE, Oakland, CA, USA, May 1998.

[23] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.

[24] D. X. Song, "Athena: a new efficient automatic checker for security protocol analysis," in *Proceedings of the 12th IEEE Computer Security Foundations Workshop*, pp. 192–202, IEEE, Mordano, Italy, June 1999.

[25] S. Escobar, C. Meadows, and J. Meseguer, "Maude-NPA: cryptographic protocol analysis mequational properties," *Foundations of Security Analysis and Design V*, vol. 5705, pp. 1–50, 2009.

[26] S. F. Doghmi, J. D. Guttman, and F. J. Thayer, "Searching for shapes in cryptographic protocols," in *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 523–537, Springer, 2007.

[27] F. Yang, S. Escobar, C. Meadows, J. Meseguer, and S. Santiago, "Strand spaces with choice via process algebra semantics," in *Proceedings of the 18th International Symposium on Principles and Practice of Declarative Programming*, pp. 76–89, ACM, New York, NY, United States, 2016.

[28] D. Basin and C. Cremers, "Know ye," *ACM Transactions on Information and System Security*, vol. 17, no. 2, pp. 1–31, 2014.

[29] X. W. Dong and W. S. Niu, "Improvement of anonymity formalization based on strand space model," *Journal on Communications*, vol. 32, no. 6, Article ID 124, 2011.

[30] P. C. Van Oorschot and M. J. Wiener, "On d-hellman key agreement with short exponents," *Advances in Cryptology - EUROCRYPT '96*, vol. 1070, pp. 332–343, 1996.

[31] C. H. Lim and P. J. Lee, "A key recovery attack on discrete log-based schemes using a prime order subgroup," *Advances in Cryptology - CRYPTO '97*, vol. 1294, pp. 249–263, 1997.

[32] L. Zhang, Y. Zhang, S. Tang, and H. Luo, "Privacy protection for e-health systems by means of dynamic authentication and three-factor key agreement," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 3, pp. 2795–2805, 2017.

[33] D. Mao, H. Liu, and W. Zhang, "An enhanced three-factor Authentication scheme with dynamic verification for medical multimedia information systems," *IEEE Access*, vol. 7, pp. 167683–167695, 2019.

[34] M. Tatebayashi, N. Matsuzaki, and D. B. Newman, "Key distribution protocol for digital mobile communication systems," in *Proceedings of the Conference on the Theory and Application of Cryptology*, pp. 324–334, Springer, Berlin, Heidelberg, July 1989.

[35] T. Matsumoto, Y. Takashima, and H. Imai, "On seeking smart public-key-distribution systems," *IEICE Transactions*, vol. 69, no. 2, pp. 99–106, 1986.

[36] H. I. Bulbul, I. Batmaz, and M. Ozel, "Wireless network security: comparison of WEP (wired equivalent privacy) mechanism, WPA (wi-fi protected access) and RSN (robust security network) security protocols," in *Proceedings of the 1st International Conference on Forensic Applications and Techniques in Telecommunications, Information, and Multimedia and Workshop*, pp. 1–6, ICST, Brussels, Belgium, Jan 2008.