Deriving Security Protocols Based on Protocol Derivation System

Ke Yang, Meihua Xiao$^{(\boxtimes)}$, Zifan Song, and Ri Ouyang

School of Software, East China Jiaotong University,
Nanchang 330013, People's Republic of China
xiaomh@ecjtu.edu.cn

Abstract. Protocol Derivation System (PDS) supports syntactic derivations of complex protocols that use cryptographic primitives. However, the PDS is only applicable for two-party interaction protocols, which does not involve in the presence of a Trusted Third Party. In this paper, we proposed an extended PDS that can support key agreement protocols using a Trusted Thirty Party by adding some components, refinements, transformations and removing redundancy rules. A flow chart of deriving security protocols based on the extended PDS is given. The flow chart consists of two layers, the first layer is to get a raw protocol using components, refinements, transformations, the second layer is to remove superfluous protocol steps and improve protocol efficiency by removing redundancy rules. Finally, we get the AOR protocol as an example to illustrate how to derive a security protocol based on the extended PDS.

Keywords: Protocol Derivation System · Security protocols · Trusted third party · Key agreement protocols · Components

1 Introduction

Security protocols are notoriously known to be difficult to get right. Designing error-free security protocols that are impervious to attack techniques, such as freshness and interleaving sessions (i.e. impersonation attacks, man-in-the-middle attacks, oracle attacks, multiplicity attacks and other types of parallel session attacks) is an extremely challenging task [1, 2]. Many protocols proposed in the literature and many protocols exploited in practice turned out to be flawed, or their well-functioning was found to be based on implicit assumptions. Such as the Needham-Schroeder authentication protocol, initiated a large body of work on the design and analysis of cryptographic protocols. In 1995, Gavin Lowe published an attack on the protocol that had apparently been undiscovered for the previous 17 years [3].

However, for security protocols, most of them are designed to meet specific application environment according to actual needs based on the intuition and experience of researchers. It turns out that designing security protocols based on experience and other informal methods has security vulnerabilities.

Formal methods have the characteristics of concise analysis process, and is recognized as an effective way to analyze the security of network protocols [4, 5]. For this reason, researchers have been working on the use of formal verification techniques to analyze the vulnerability of security protocols. A large number of formal methods have been proposed, which including modal logic, theorem proving and model checking [6–8].

In 2005, a framework (denoted DDMP framework) consisting of Protocol Derivation System (PDS) and Protocol Composition Logic (PCL) [9, 10] has been proposed by Datta et al. PDS supports syntactic derivations of complex protocols, starting from basic components, and combining or extending them using a sequence of composition, refinement, and transformation operations [11]. PCL is a Floyd-Hoare style logic that supports axiomatic proofs of protocol properties. The eventual goal is to develop proof methods for PCL for every derivation operation in PDS, thereby enabling the parallel development of protocols and their security proofs. Since the DDMP framework was proposed, this method has been successfully applied to the formal analysis and security proof of many protocols. Datta et al. described the deduction process of various protocols in STS family, and successfully analyzed ISO-9798-3. Roy et al. [12] extended the original PCL for the confidentiality of security protocol. Mitchell et al. [13] have successfully analyzed industrial protocols such as SSL/TLS and Kerberos V5 using PCL. Li et al. [14] proposed a PCL secure user authentication protocol named UCAP for cloud computing. Zhang Junwei et al. [15] proposed a PCL secure and efficient group authentication protocol named TSNP. PCL is an effective way to provide formal proofs to the correctness of security protocols.

There is no doubt that DDMP framework has been successfully applied to a number of industrial network security protocols. However, there are only a little improvement on the PCL rather than PDS after the DDMP framework.

PDS for deriving security protocols consists of a set of basic building blocks named components and a set of operations for constructing new protocols from old ones. These operations can be divided into three different types: composition, refinement and transformation. A component is used as a building block for larger protocols. For example, the Diffie-Hellman key exchange and challenge-response component are basic components providing the desired security properties. The composition operation combines two sub-protocols. Parallel composition and sequential composition are two composition operations. The refinement operation acts on message components of a single protocol. The original PDS mainly aims at the design of two-party interaction protocols including initiator and responder, without the existence of a Trusted Third Party. In addition, the main set of operations for PDS is public key encryption [16]. Although Zhang Junwei et al. [17–19] added some new components, refinements and transformation to the PDS, their work mainly aims to derive the key exchange protocols in the Needham-Schroeder family. Therefore, in order to derive key agreement protocols using a Trusted Third Party in symmetric encryption environment, we intend to extend the PDS.

Our Contributions. In this paper, we study the extension of PDS to support the derivation of key agreement protocols using a Trusted Third Party in symmetric encryption environment.

1. Some components, refinements, transformations and removing redundancy rules are added to the PDS. The extended PDS can carry out protocol design using combination method based on symmetric encryption environment.
2. A flow chart of deriving security protocols based on the extended PDS is given. The flow chart consists of two layers, the first layer is to get a raw protocol using components, refinements, transformations, the second layer is to remove superfluous protocol steps and improve protocol efficiency.
3. Deriving a key agreement protocol in symmetric environment is given as an example to illustrate how the extended PDS supports the derivation of security protocols.

The rest of this paper is organized as follows. Section 2 introduces some preliminaries about PDS and the defects of PDS. Section 3 gives the extension of PDS. In Sect. 4, we give the detailed derivation of security protocols to illustrate how to derive protocols based on the extended PDS. Section 5 concludes.

2 Protocol Derivation System and Its Defects

PDS is a useful method to design and generate security protocols syntactically, which consists of a set of basic building blocks called components and a set of operations for constructing new protocols from old ones. First, we give brief introduction about components, operations and transformations, then the defects in PDS are presented.

2.1 The Explanation of Symbols

A, B, S:	Initiator, Responder and Server (Trusted Thirty Party).
K:	shared key. In symmetric encryption environment, the encryption key is the same as the decryption key, which can be written as $K = K^{-1}$.
N_a, N_b:	random numbers generated by *A* and *B*.
m:	A message.
K_{xy}:	the shared key K_{xy} between *X* and *Y*.
$SIG_X(m)$:	A signature of *m* created *by X*.
$X \rightarrow Y$:	*X* sends a message *m* to *Y*.
$p \Rightarrow q$:	block *p* is replaced by block *q*

2.2 Components

A protocol component consists of a set of roles (e.g., initiator, responder, server), where each role has a sequence of inputs, outputs and protocol actions. Diffie-Hellman key exchange and signature-based authenticator are basic components in PDS.

Diffie-Hellman Component, C_1
The Diffie-Hellman protocol [20] provides a way for two parties to set up a shared key (g^{ir}) which a passive attacker cannot recover. There is no authentication guarantee: the secret is shared between two parties, but neither can be sure of the identity of the other. The initiator and responder role actions are given below.

I: generates random value *i* and computes g^i (for previously agreed base *b*)
R: generates random value *r* and computes g^r (for previously agreed base *b*)

In this component no messages are sent.

Signature-Based Authenticator, C_2
The signature-based challenge-response protocol shown below is a standard mechanism for one-way authentication.

$I \rightarrow R$: *m*
$R \rightarrow I$: $SIG_r\ (m)$

It is assumed that *m* is a fresh value or nonce and that the initiator, *I*, possesses the public key certificate of responder, *R*, and can therefore verify the signature.

2.3 Composition

The composition operation used is sequential composition of two protocol components with term substitution. The roles of the composed protocol have the following structure: the input sequence is the same as that of the first component and the output is the same as that of the second component; the actions are obtained by concatenating the actions of the first component with those of the second (sequential composition) with an appropriate term substitution-the outputs of the first component are substituted for the inputs of the second.

2.4 Refinements

Refinement R_1 $SIG_X\ (m) \Rightarrow EK\ (SIG_X\ (m))$, where *K* is a key shared with the peer. The purpose of this refinement is to provide identity protection against passive attackers
Refinement R_2 $SIG_X\ (m) \Rightarrow SIG_X\ (HMAC_K\ (mID_X))$, where *K* is a key shared with the peer. While the signature by itself proves that this term was generated by *X*, the keyed hash in addition proves that *X* possesses the key *K*.
Refinement R_3 $SIG_X\ (m) \Rightarrow SIG_X\ (m),\ HMACK\ (m,\ ID_X)$, where *K* is a key shared with the peer. This refinement serves the same purpose as R_2.
Refinement *R4* $SIG_X\ (m) \Rightarrow SIG_X\ (mID_Y)$, where *Y* is the peer. It is assumed that *X* possesses the requisite identifying information for *Y*, e.g., *Y's* public key certificate, before the protocol is executed. This assumption can be discharged if *X* receives *Y's* identity in an earlier message of the protocol.
Refinement *R5* $g^x \Rightarrow g^x,\ n^x$, where n^x is a fresh value. The use of nonce enables the reuse of exponentials across multiple sessions resulting in a more efficient protocol.
Refinement R_6 $SIG_X\ (m) \Rightarrow SIG_X\ (m),\ ID_X$, where ID_X denotes the public key certificate of *X*. This refinement enables that the principals possess each other's public key certificates before the session.
Refinement R7 $E_K\ (m) \Rightarrow E_K\ (m),\ HMAC\ _{K'}\ (role,\ E_K\ (m))$, where *K* and *K'* are keys shared with the peer and role identifies the protocol role in which this term was produced (initiator or responder).

2.5 Transformations

Message Component Move, T_1

This transformation moves a top-level field *t* of a message *m* to an earlier message *m'*, where *m* and *m'* have the same sender and receiver, and if *t* does not contain any data freshly generated or received between the two messages. One reason for using this transformation is to reduce the total number of messages in the protocol.

Binding, T_2

Binding transformations generally add information from one part of a protocol to another in order to "bind" the two parts in some meaningful way. The specific instance of this general concept that we use here adds a nonce from an earlier message into the signed portion of a later message, this operation is illustrated as follows.

$$\begin{array}{lcl} I \rightarrow R: m & & I \rightarrow R: m \\ R \rightarrow I: n,\ SIG_R\,(m) & \Longrightarrow & R \rightarrow I: n,\ SIG_R\,(m,\ n) \\ I \rightarrow R: SIG_I\,(n) & & I \rightarrow R: SIG_I\,(m,\ n) \end{array}$$

Cookie, T_3

The purpose of the cookie transformation is to make a protocol resistant to blind Denial-of-Service (DoS) attacks [21]. Under certain assumptions, it guarantees that the responder does not have to create state or perform expensive computation before a round-trip communication is established with the initiator.

$$\begin{array}{lcl} I \rightarrow R: m_1 & & I \rightarrow R: m_1 \\ R \rightarrow I: m_2 & \Longrightarrow & R \rightarrow I: m_2^c,\ HMAC_{HKR}\,(m_1,\ m_2^c) \\ I \rightarrow R: m_3 & & I \rightarrow R: m_3,\ m_1,\ m_2^c\ ,\ HMC_{HKR}\,(m_1,\ m_2^c) \\ & & R \rightarrow I: m_2^e \end{array}$$

To sum up, key agreement protocols using a Trusted Thirty Party differ from the STS family and Needham-Schroeder family in the following characteristics.

1. Involvement of a Trusted Third Party. There are only the Diffie-Hellman component and signature-based authenticator component in PDS, which can't derive key agreement protocols using a Trusted Thirty Party
2. Symmetric encryption. Due to the lack of components and operations to describe this kind of encryption type in PDS, symmetric key exchange protocol can't be derived by PDS.
3. More interactive steps. Because of the involvement of Trusted Third Party and symmetric encryption-based authentication, it's necessary to remove superfluous protocol steps and improve protocol efficiency by removing redundancy rules.

3 The Extension for PDS

For the original PDS exists some defects, an extended PDS is proposed. Not only some components and transformations are added in the extended PDS, but also some removing redundancy rules which don't exist in original PDS are proposed, the purpose of these rules is to reduce superfluous protocol steps and improve efficiency.

3.1 Extended Components

Timestamp Component, C_3

$A \rightarrow S$: $\{A, B\}K_{as}$
$S \rightarrow A$: $(A, M, T_S)K_{as}, (B, M, T_S)K_{bs}$

The client requests from the server. If the request information is encrypted and intercepted by a third party to the request package, the request package can be used for repeated requests. If the server does not prevent replay attack, the server pressure will increase, and the use of timestamp can solve this problem.

Trusted Thirty Party Component, C_4

$A \rightarrow S$: A, B
$S \rightarrow A$: K_{ab}
$A \rightarrow B$: K_{ab}

This component is a basic framework for key agreement protocols with Trusted Thirty Party. In this component, the initiator A sends his identity and B's identity to the server. Then the server responds with shared key K_{ab} between A and B. Finally, the initiator sends K_{ab} to B. This component can provide key agreement but don't ensure authentication and confidentiality.

Challenge-Response Component, C_5

$C \rightarrow R$: N_C
$R \rightarrow C$: $E_K\{m, N_C\}$

In contrast to signature-based authenticator component C_2, this component is encrypted by symmetric key and the request information is a random number rather than message *m*. Using a series of Challenge-Response steps, which can provide user authentication to some degree.

Three-Party Authentication Component, C_6

$A \rightarrow B$: $A, \{A, N_a\} K_{as}$
$B \rightarrow S$: $\{N_b, B\}K_{bs}, \{A, N_a\} K_{as}$
$S \rightarrow B$: $\{Na, N_b\}K_{bs}$

The three-party authentication component C_6 is a basic framework for key agreement protocols using a Trusted Third Party. Because of the existence of N_a and N_b, This Component can ensure the authentication between *A* and *B* through a Trusted Thirty Party.

3.2 Extended Refinements

Refinements are mainly for the cryptographic operation of messages in the protocol. Without changing the number of messages or the structure of the protocol, the necessary security attributes can be added to a message component by refinements.

There are seven refinements R_1–R_7 in the existing PDS system, which are mainly for public key cryptosystem. The cryptographic refinement operations based on symmetric key encryption are as follows.

Refinement R_8: $m \rightarrow E_K(m)$. The message *m* is encrypted by symmetric key *K*, no one can decrypt message $E_K(m)$ unless the agent discloses its own private key. This refinement can guarantee the confidentiality of message *m* to some degree.
Refinement R_9: $E_K(m) \rightarrow E_K(m, X)$. The identity of *X* is added to $E_K(m)$, therefore this refinement can prevent man-in-middle attack effectively.
Refinement R_{10}: $E_K\{N_a\} \rightarrow E_K(N_a-1)$. Responding the message $E_K(N_a-1)$ rather than $E_K\{N_a\}$, which can refresh the random number N_a and prevent reply attack effectively.
Refinement R_{11}: $E_K(X) \rightarrow E_K(X, N_a)$. By adding the random number N_a, this refinement can prevent reply attack to some degree because of the freshness of N_a

3.3 Extended Transformations

Symmetry, T_8

$A \rightarrow B$: $X, A, B, \{N_a, A\}K_{as}$	$\Longrightarrow$	$A \rightarrow B$: $X, A, B, \{N_a, A, B\}K_{as}$
$B \rightarrow S$: $X, A, B, \{N_a, A\}K_{as}, \{N_b, B\}K_{bs}$		$B \rightarrow S$: $X, A, B, \{Na, A\}Kas, \{N_b, B\}K_{bs}$

Changing Order, *T_9*

$$\begin{array}{lcl}
A \to S: A, B & & A \to B: A \\
S \to A: \{K_{ab}, B\}K_{as}, \{K_{ab}, A\}K_{bs} & \Longrightarrow & B \to A: A, B \\
A \to B: \{K_{ab}, A\}K_{b} & & S \to B: \{K_{ab}, A\}K_{bs}, \{K_{ab}, B\}K_{as} \\
 & & B \to A: \{K_{ab}, B\}K_{as}
\end{array}$$

Index, *T_{10}*
To denote a run of protocol, we add a unique index to every step of transformed protocol.

$$\begin{array}{lcl}
A \to B: A, B & & A \to B: I, A, B \\
B \to S: A, B & \Longrightarrow & B \to S: I, A, B \\
S \to B: \{K_{ab}\}K_{bs}, \{K_{ab}\}K_{as} & & S \to B: I, \{K_{ab}\}K_{bs}, \{K_{ab}\}K_{as} \\
B \to A: \{K_{ab}\}K_{as} & & B \to A: I, \{K_{ab}\}K_{as}
\end{array}$$

Consistency, *T_{11}*
Sometimes we need to ensure the consistency of shared key through key agreement protocols, so the transformation T_{11} is proposed.

$$\begin{array}{lcl}
A \to B: A, B & & A \to B: A, B \\
B \to S: A, B & \Longrightarrow & B \to S: A, B \\
S \to B: \{K_{ab}\}K_{bs}, \{K_{ab}\}K_{as} & & S \to B: \{K_{ab}\}K_{bs}, \{K_{ab}\}K_{as} \\
B \to A: \{K_{ab}\}K_{as} & & B \to A: \{K_{ab}\}K_{as}, \{B\}K_{ab}
\end{array}$$

3.4 Removing Redundancy Rules

There are some redundant operations and items in redundancy protocol, which leads to redundant authentication and affects efficiency. At the same time, the operation items in the security protocol are too long, which will affect the efficiency, especially when both agents use public key to achieve security properties. Therefore, we introduce the following removing redundancy rules.

Cascade Connection, *RR_1*
If there are multiple encryption structures in the protocol step *i*, and these encryption structures use the same cryptographic algorithm and key, thus these can be combined into one structure. The cascade connection is expressed in logical language as follows:

$$\begin{array}{lcl}
B \text{ receive } E_K(m_1) & & \\
 & \Longrightarrow & B \text{ receive } E_K(m_1, m_2) \\
B \text{ receive } E_K(m_2) & &
\end{array}$$

Deleting Duplicate Information, *RR_2*
If the same message appears twice in a protocol step, only one of them is retained. The deleting duplicate information is expressed in logical language as follows:

$$\begin{array}{lcl} B\ receive\ E_K(m) & & \\ & \Rightarrow & B\ receive\ E_K(m) \\ B\ receive\ E_K(m) & & \end{array}$$

Freshness, *RR_3*
If the data items used to prove freshness in protocol step *i* (such as random numbers) always appear at the same time as a data item with freshness in the protocol, then the data items used to prove freshness can be deleted. The rule is expressed in logical language as follows:

$$\begin{array}{lcl} Has(X, MN) \wedge Fresh(MN) & & \\ & \Rightarrow & Has(X, MN) \wedge Fresh(MN) \\ Has(X, M) \wedge Fresh(M) & & \end{array}$$

Plaintext, *RR_4*
If a plaintext *m* appears in the encryption part of the protocol, the plaintext *m* is encrypted first and then simplified by cascade connection RR_1.

4 Deriving Security Protocols Based on the Extended PDS

Based on the extended PDS, we give a flow chart to derive security protocols. This flow chart can be divided into two layers. The first layer is to get a raw protocol meeting expected security properties. First, combining the basic cryptographic primitives, we can get some components, which are the basic framework for different kinds of security protocols. According to expected security properties, for example, the mutual authentication or the key agreement between initiator and responder, choosing some correct components and then getting a single step protocol. Next, based on the composition rules, appropriate refinements and transformations, a raw protocol can be concluded. The second layer is to remove redundancy and improve efficiency. Through removing redundancy rules, a target protocol can be derived. The above steps are illustrated in Fig. 1.

In order to better illustrate how to derive a security protocol based on the extended PDS, we will derive a three-party key agreement protocol as a case study.

1. First, our objective protocol is a three-party key agreement protocol, so we choose C_4 as the basic component, which provides key agreement using a Trusted Thirty Party.
2. Protocol P_1 obtained by applying R_8 to C_4.

$A \rightarrow S:\ A, B$

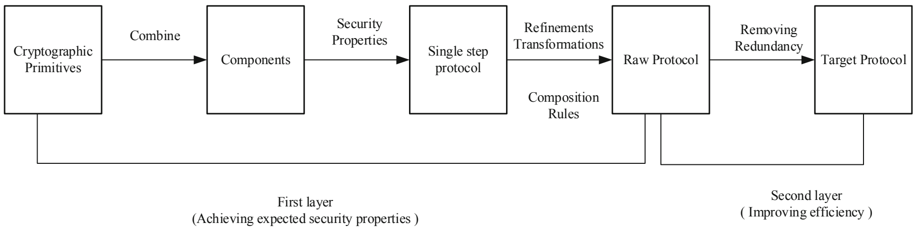

Fig. 1. The flow chart of deriving security protocols based on the extended PDS

$S \rightarrow A$: $\{K_{ab}\}K_{as}$, $\{K_{ab}\}K_{bs}$
$A \rightarrow B$: $\{K_{ab}\}K_{bs}$

3. Protocol P_2 obtained by applying T_9 to P_1.

$A \rightarrow B$: A
$B \rightarrow S$: A, B
$S \rightarrow B$: $\{K_{ab}\}K_{as}$, $\{K_{ab}\}K_{bs}$
$B \rightarrow A$: $\{K_{ab}\}K_{as}$

4. Protocol P_3 obtained by applying T_{10} to P_2.

$A \rightarrow B$: M, A
$B \rightarrow S$: M, A, B
$S \rightarrow B$: M, $\{K_{ab}\}K_{as}$, $\{K_{ab}\}K_{bs}$
$B \rightarrow A$: M, $\{K_{ab}\}K_{as}$

5. The goal of key agreement can be achieved by protocol P_3 basically. Now we need to add component C_6 which provides mutual authentication between initiator and responder. Protocol P_4 obtained by applying C_6 to P_3.

$A \rightarrow B$: $M, A, \{N_a, A\}K_{as}$
$B \rightarrow S$: $M, A, B, \{N_a, A\}K_{as}, \{N_b, B\}K_{bs}$
$S \rightarrow B$: $M, \{K_{ab}\}K_{as}, \{N_a, A\}K_{as}, \{K_{ab}\}K_{bs}, \{N_b, B\}K_{bs}$
$B \rightarrow A$: $M, \{K_{ab}\}K_{as}, \{N_a, A\}K_{as}$

6. We can learn that some messages are encrypted by the shared key, It's necessary to remove superfluous protocol steps and improve protocol efficiency. So protocol P_5 obtained by applying cascade connection RD_1 to P_4.

$A \rightarrow B$: $M, A, \{N_a, A\}K_{as}$
$B \rightarrow S$: $M, A, B, \{N_a, A\}K_{as}, \{N_b, B\}K_{bs}$
$S \rightarrow B$: $M, \{K_{ab}, N_a, A\}K_{as}, \{K_{ab}, N_b, B\}K_{bs}$
$B \rightarrow A$: $M, \{K_{ab}, N_a, A\}K_{as}$

7. Protocol P_6 obtained by composition P_5 and T_8.

$A \rightarrow B$: $M, A, B, \{N_a, M, A, B\}K_{as}$
$B \rightarrow S$: $M, A, B, \{N_a, M, A, B\}K_{as}, \{M, N_b, A, B\}K_{bs}$
$S \rightarrow B$: $M, \{K_{ab}, N_a, N_b, A\}K_{as}, \{K_{ab}, N_b, B\}K_{bs}$
$B \rightarrow A$: $M, \{K_{ab}, N_a, N_b, A\}K_{as}$

8. Protocol P_7 obtained by applying T_{11} to P_6.

$A \rightarrow B$: $M, A, B, \{N_a, M, A, B\}K_{as}$
$B \rightarrow S$: $M, A, B, \{N_a, M, A, B\}K_{as}, \{M, N_b, A, B\}K_{bs}$
$S \rightarrow B$: $M, \{K_{ab}, N_a, N_b, A\}K_{as}, \{K_{ab}, N_b, B\}K_{bs}$
$B \rightarrow A$: $M, \{K_{ab}, N_a, N_b, A\}K_{as}, \{N_b\}K_{ab}$

These above steps are shown in a flow chart Fig. 2.

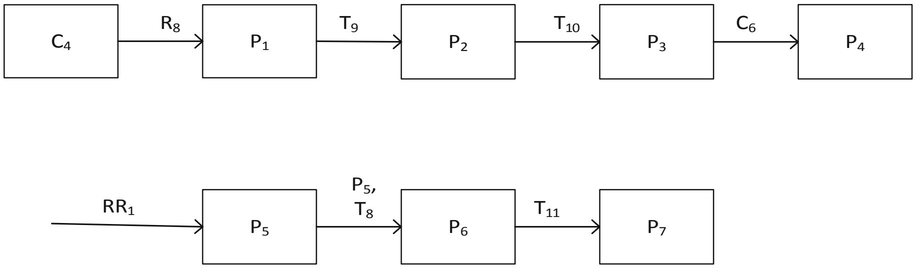

Fig. 2. Deriving three-party key agreement protocol based on the extended PDS

The final protocol P_7 is an AOR protocol, whose security properties have been proved by PCL in reference [22].

Through the combinations and refinements of the basic components, the target security protocol is obtained. Compared with a security protocol based on expert intuition and experience, the target security protocol obtained in this paper is more secure to a certain extent. Of course, to further ensure the security properties the target protocol, we can provide logical proof for the security protocols based on PCL.

In additional, in order to illustrate the advantage of the extended PDS, we give an overall comparison with Datta [9] and Zhang Junwei's methods [17, 18].

	Trusted Third Party	Symmetric Encryption	Removing Redundancy Rules	Timestamp	Three-Party Authentication
Datta	×	×	×	×	×
Zhang Junwei	√	√	×	√	×
This paper	√	√	√	√	√

Compared with Datta and Zhang Junwei's methods, the main advantages of this paper add a Trusted Third Party, removing redundancy rules and three-party authentication component, which support the derivation of public key or symmetric key protocols with trusted third party. In addition, some removing redundancy rules are added in the PDS in order to reduce redundant information and improve efficiency.

5 Conclusions

PDS is a useful method to design and generate security protocols syntactically. However, the PDS is only applicable for two-party interaction protocols, which does not involve in the presence of a Trusted Third Party. In this paper, we proposed an extended PDS that can support key agreement protocols using a Trusted Thirty Party by adding some components, refinements, transformations and removing redundancy rules. A flow chart of deriving security protocols based on the extended PDS is given. The flow chart consists of two layers, the first layer is to get a raw protocol using components, refinements, transformations, the second layer is to remove superfluous protocol steps and improve protocol efficiency by removing redundancy rules. We get the improved version of Otway-Rees protocol as an example to illustrate how to derive a security protocol based on the extended PDS.

In DDMP framework designed by Datta, PCL can provide logical proof for the security protocols generated by PDS. But at present, PCL can't support the logical proof of every step in the PDS system. In additional, deriving different types of security protocols supported by PDS is still our future work.

Acknowledgements. This work is supported by following projects: The National Natural Science Foundation of China under Grant No.61962020 and No.61562026, the Jiangxi Province Key Subject Academic and Technical Leader Funding Project under Grant No. 2017XSDTR0105.

References

1. Sihan, Q.: Twenty years development of security protocols research. J. Software **14**(10), 1740–1752 (2000)
2. Yong, L., Fan, Z., Ming, Z.: Survey on security protocol space information network. Comput. Sci. **44**(04), 202–206 (2017)
3. Lowe, G.: Breaking and fixing the Needham-Schroder public-key protocol using FDR. In: Proceeding of TACAS, LNCS 1055, Spring, pp. 147–166 (1996)
4. Avalle, M., Pironti, A., Sisto, R.: Formal verification of security protocol implementations: a survey. Formal Aspects of Comput. **26**(1), 99–123 (2014)
5. Jian, W., Naijun, Z., Fen, X., Liu, Z.: Overview of formal methods. J. Software, **30**(01), 33–61 (2019)
6. Agray, N., Wiebe, V.D.H., De Vink, E.: On BAN logics for industrial security protocols. In: International Workshop of Central and Eastern Europe on Multi-Agent Systems. Springer, Berlin (2001)
7. Hess, A.V., Sebastian, M.: Formalizing and proving a typing result for security protocols in Isabelle/HOL. In: 2017 IEEE 30th Computer Security Foundations Symposium (CSF). IEEE (2017)
8. Basin, D., Cremers, C., Meadows, C.: Handbook of Model Checking- Model Checking Security Protocols, pp. 727–762. Springer, Cham (2018)

9. Datta, A., Derek, A., Mitchell, J.C., Roy, A.: "Protocol Composition Logic (PCL)", Electronic Notes in Theoretical Computer Science. Gordon D, Plotkin Festschrift (2007)
10. Derek, A.: Formal analysis of security protocols: protocol composition logic, Ph.D. Dissertation, Stanford University (2007)
11. Datta, A., Derek, A., Mitchell, J.C., Pavlovic, D.: A derivation system and compositional logic for security protocols. J. Comput. Security **13**(3), 423–482 (2005)
12. Roy, A., Datta, A., Derek, A., Mitchell, J.C., Seifert, J.-P.: Secrecy analysis in protocol composition logic. In: Formal Logical Methods for System Security and Correctness, IOS Press (2008)
13. Datta, A., Anupam, et al.: Computationally sound compositional logic for key exchange protocols. In: 19th IEEE Computer Security Foundations Workshop (CSFW 2006), IEEE (2006)
14. Li, X., Zhang, J., Ma, J.: UCAP: a PCL secure user authentication protocol in cloud computing. J. Commun. **39**(08), 94–105 (2018)
15. Li, X., Zhang, J., Ma, J., Hai, L.: TSNP: a novel PCL-Secure and efficient group authentication protocol in space information network. J. Comput. Res. Dev. **53**(10), 2376–2392 (2016)
16. Cremers, C.: On the protocol composition logic PCL. In: Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security (2008)
17. Zhang, J., Yang, C., Ma, J.: Protocol derivation system for the Needham-Schroeder family. In: 2011 6th International ICST Conference on Communications and Networking in China (CHINACOM), pp. 836–840 (2011)
18. Zhang, J., Ma, J., Chao, Y.: Protocol derivation system for the needham-schroeder family. Secur. Commun. Netw. **8**, 2687–2703 (2015)
19. Lu, L.: Study on theory and applications of security protocols formal analysis, Ph.D. Dissertation, Xidian University (2012)
20. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Trans. Inf. Theory, IT-**22**(6), 644–654 (1976)
21. Datta, A., Mitchell, J.C., Pavlovic, D.: Derivation of the JFK protocol, Technical Report KES.U.02.03, Kestrel Institute (2002)
22. Lu, L., Duan, X., Ma, J.: Improvement and formal proof on protocol Otway-Rees. J. Commun. **33**(1), 250–254 (2012)