A Weakest Precondition Calculus for Analysis of Cryptographic Protocols*

J. Alves-Foss and T. Soule
Laboratory for Applied Logic
Department of Computer Science
University of Idaho
{jimaf,tsoule}@cs.uidaho.edu

Abstract

This paper present a new approach for the use of belief logics in the analysis of cryptographic protocols. This new approach is based on the concept of the weakest precondition calculus. It generates the necessary preconditions of a protocol, based on the protocol operations and desired goals. In addition, the approach we use overcomes some conceptual problems of belief logic analysis by forcing explicit denotation of internal protocol participant operation. Complete analysis rules and methods are presented, along with exemplary applications. The paper concludes with a discussion of the use of this approach for protocol derivation.

1 Introduction

The use of formal methods to analyze the correctness of cryptographic protocols became prevalent with the development of the BAN logic in 1989 [BAN89]. In this style of analysis a set of participant's final *beliefs* is generated from a set of initial assumptions and the protocol messages. If these beliefs satisfy the goal of the protocol, then the protocol is validated. Since the publication of the BAN logic, several papers have reported problems in its use for analysis of cryptographic protocols, including [Mea96, Sne91, Syv93]. These reports reveal limitations of the logic or misunderstanding and misuse of the logic. Descendents of the BAN logic such as GNY [GNY90] and SvO [SvO94], called BAN-style logics, overcome several of these limitations. A recent paper, by one of the authors of this paper, demonstrates that the published failures of belief logic analysis has been due to the improper use of the logics and not due to limitations of the logics [AF97]. This fact is also discussed in [SvO96]. In [AF97], we suggest a more formalized approach to cryptographic protocol analysis using BAN-style logics. This approach reduces the likelihood of improper use of the logic.

This paper extends our other work by developing a *weakest-precondition* (wp) calculus for belief logic analysis of cryptographic protocols. This calculus, based on the SvO logic [SvO94], provides a

*This project was sponsored in part by the National Security Agency under Grant Number MDA904-96-1-0108. The United States Government is authorized to reproduce and distribute reprints notwithstanding any copyright notation hereon.

methodical approach to the specification and analysis of cryptographic protocols. The application of the techniques outlined in this paper provides users with a clear path toward protocol validation, avoiding many of the pitfalls of the past. The development of a wp-calculus for authentication protocols also lends itself toward automating the development of secure protocols given specific system requirements, using derivational programming techniques.

Specifically, we have classified the actions performed by participants of a cryptographic protocol into eight different operations (composition, message sending, message receiving, function application, key creation, shared secret creation, message decryption and proving). For each of these operations we define not only the general wps but also specific preconditions for each of the possible predicates defined by the semantics (e.g., believing, freshness, seeing, etc.). This separate treatment of each predicate is useful given the semantic nature of the operations and the belief logics, and provides users with precise guidelines for the application of the logics. In this paper we will present the wp-calculus, the specific preconditions defined for each operation, discuss how to use the wp-calculus and demonstrate its use on exemplary protocols.

2 Analysis of Authentication Protocols

When using belief logics for protocol validation, the typical approach has been to establish a set of premises or preconditions; generate beliefs by using the steps of the protocol and the belief logic [BAN89, GNY90, SvO94, SvO96]. Although the exact format of the protocol and the rules of the logic may differ, the approach is basically the same. In Syverson and van Oorschot's most recent paper [SvO96] they present a set of common premises and postconditions that can be used in protocol analysis. In addition, the suggest adding premises for all messages such that upon receipt of a message with uncomprehended component, we can assign a meaning to some of those components — this is in lieu of protocol idealization which was used in earlier papers, and sometimes resulted in unintentional modification of the semantics of the protocol. There are several issues in the standard approach which need to be considered.

- **Premises**. All proofs require a set of premises/preconditions that must hold prior to protocol execution. Generation of these premises, although often standard, requires some thought. It is possible to generate too many premises. Our approach automatically generates these premises.
- **Message Transmission/Receipt**. The standard approach to validation assumes that all messages have already been received. It is possible to utilize the premises and messages received to derive incorrect or faulty beliefs. This is due to the fact that message receipt is not ordered [Sne91, AF97]. Our approach insists that message reception be treated in order.
- **Idealization/Comprehension Premises**. In any protocol, there are certain assumptions that the recipients make with respect to message meaning. These can be expressed in idealized versions of the message protocols, or in premises; but they must be expressed for the analysis to complete. If these premises or idealizations are incorrect, the analysis will fail. In [AF97] we provide a set of guidelines for the development of these idealizations; which are used in the analysis in this paper.

- **Verification**. It is often the case that the beliefs and comprehended messages of both participants are analyzed concurrently[1]. This may result in the use of incorrect premises during the analysis. It also results in a more complicated proof. The technique we present here completely separates the proofs of the participants. This is also useful, since it does not force us to assume specific actions of the other participants; only that a message of a particular form has been received.

- **Postconditions**. The ultimate goal of protocol verification is to guarantee that following completion of the protocol, certain beliefs hold. The common protocol goals of [SvO96] are a good starting point for generation of these goals. These goals should be consistent with the protocol designer's intentions.

- **Protocol Denotation**. Most verification methods utilize the standard notation for protocol denotation. This involves the sequential list of all protocol messages and direction sent. Any auxiliary operations such as function application, encryption, decryption, etc. are only described textually. Our approach forces the explicit denotation of all of these steps for a more precise understanding of the protocol.

In particular, we present an approach to protocol verification based on the concept of the weakest precondition. The idea is to generate a set of postconditions/goals that we want the protocol to satisfy. Then, based on these postconditions and the steps of the protocol, generate the least restrictive set of preconditions such that upon completion of the protocol the goals will hold. To simplify the analysis, we handle each step of the protocol one at a time, propagating the generated preconditions through each step. To simplify presentation of this approach we assume that all conditions (pre and post) are represented in a set of conditions. The tables presented in the next section do not specifically reference the set, but rather only particular elements of the set. In other words, if a condition is relevant to a particular protocol step, it will be removed from the set of conditions and replaced with the necessary preconditions, otherwise it will be automatically propagated through as a precondition.

The choices regarding what the postconditions of an operation should be and what preconditions are necessary for those postconditions were primarily determined by the SvO logic [SvO94, SvO96]. The operations currently supported are: sending and receiving messages, applying functions, creating keys from public agreement keys, decrypting messages, construction of shared secrets, and proving statements. This includes the use of private, public, and shared keys. The wps were tested on several existing protocols. A discussion of the proof for the station-to-station protocol [DvOW92] and the Needham-Schroeder key distribution protocol [NS78] are included in Section 4 of this document.

Section 3 presents the weakest precondition rules. They are divided into four subsections, one for the general operations, one for proofs and beliefs, one for decryption using the various key types, and one for construction shared secrets. Section 4 presents analysis of the two exemplary protocols. Section 5 discusses a few issues regarding the use of the wps.

[1] In [SvO96], Syverson and van Oorschot determine final beliefs for each participant independently, but combine the premises for the participants.

3 Weakest Preconditions

This section describes the wps for each of the authentication operations. In general, a wp acts as a textual substitution which replaces one condition with another set of equivalent conditions. The wps for each operation are presented in tables of three columns: the postcondition, the weakest precondition necessary for the postcondition, and the reasoning. Axiom numbers refer to SvO axioms in [SvO94].

The wps depend in part upon which postconditions are desired. Some postconditions are true, regardless of the preconditions, as long as the operation is performed. For example, the operation "B receives X" makes the postconditions "B receives X" and "B sees X" automatically true. Similarly, some preconditions are required regardless of what postconditions are desired. For example, the operation "B sends X" has as a required precondition "A sees X" which must be true. These required new weakest preconditions are marked with a star.

Many of the weakest preconditions have corresponding conditions regarding the beliefs of the active party. The corresponding conditions for belief are derived using the same axiom as the original substitution plus necessitation. For example, the weakest preconditions of "fresh(X)" are "fresh(X_i) $\wedge$ $X_i \in X$" by Axiom 17 and thus for beliefs the weakest preconditions of "B believes (fresh(X))" are "B believes (fresh(X_i)) $\wedge$ $X_i \in X$", by Axiom 17 and necessitation. Note that necessitation may only be used in conjunction with theorems and not derivations [SvO96].

3.1 General Operations

This section provides the wps for the five general operations of an authentication protocol:

- $S;T$ {Composition}
- $\rightarrow B : X$ {B receives X}
- $A \rightarrow: X$ {A sends X}
- $A : F(X,Y) :$ {A applies function F to X and Y}
- $A\ CreateKey$: K {A creates a key from public agreement keys}

The first three operations listed are explicitly denoted in the presentation of any authentication protocol. They involve the sending and receiving of messages as well as a sequential ordering of operations. It is interesting to note that most protocol specifications combine the act of sending and receiving messages as a single operation. This is acceptable if we can be assured that both parties are following the correct operations of the protocol, and are who they claim to be. As such, we prefer to distinguish between the acts of sending and receiving messages, to enable us to evaluate the actions and beliefs of the participants independently.

In Table 1 we present the wp for operation composition. This enables us to treat each operation separately by treating the wp of operation T as the postcondition to operation S.

To enable communication between participants we have the operations of sending and receiving messages. In Table 2 we present the wps for message sending. The semantics here state that prior

Table 1: Weakest preconditions for composition of two operations.
Operation: $S;T$

Old Postcondition	Weakest Precondition	Reasoning
wp.(S;T).R	wp.S.(wp.T.R)	standard definition

Table 2: Weakest preconditions for sending a message.
Operation: $A \rightarrow: X$

Old Postcondition	Weakest Precondition	Reasoning
	* A sees X	Semantics
A said X A sent X	True	Semantics Semantics
A said X_i	A sees X_i $X_i \in X$	Semantics

* Necessary weakest precondition.

to sending a message a participant must have seen the message. In addition, if we want to state that a participant sent a component of a message, then it is necessary that they see the component. In other words, we can not claim that A sent X_i if A only passed along a full uncomprehended message that contained X_i. It is also worth noting that if we limit the evaluation of the protocol to only what can be inferred directly by the recipient (as discussed by Syverson and van Oorschot in [Syv96, SvO96]), we never need worry about messages sent by another participant. We leave this wp in place however, since there may be analysis methods that require that a participant be aware of messages previously sent by itself (see [AF97]).

In Table 3 we present the wps for message receipt. The conditions in this case are rather straight forward. Upon receipt of a message X, B will have received it and seen it. B will also have received and seen all components of the message. If we follow the message comprehension method of Syverson and van Oorschot [SvO96], B receives and sees the uncomprehended messages $*_i$. Additional preconditions must be put in place to specify what B comprehends; work on this is left for a future paper. The message will be considered fresh if there is a component of X that is fresh.

Table 3: Weakest preconditions for receiving a message.
Operation: $\rightarrow B : X$

Old Postcondition	Weakest Precondition	Reasoning
B received X B sees X	True	Semantics Axiom 10
B received X_i B sees X_i	$X_i \in X$	Axiom 7 Axiom 11
fresh(X)	$\exists X_i$:$X_i \in X$ fresh(X_i)	Axiom 17

* Necessary weakest precondition.

Table 4: Weakest preconditions for applying a function.
Operation: A: applies $F(X_1, \ldots X_N)$

Old Postcondition	Weakest Precondition	Reasoning
	* $\forall\ i : 1 \ldots N$. A sees X_i	Axiom 12
A sees $F(X_1, \ldots, X_N)$	True	Axiom 12
fresh($F(X_1, \ldots, X_N)$)	$\exists\ i \in 1 \ldots N$: fresh(X_i)	Axiom 17 and 18

* Necessary weakest precondition.

Function application is one of the most basic operations of an authentication protocol, and is usually not explicitly stated in the presentation of a protocol but rather only in the textual description of the protocol. This includes, but is not limited to encryption, decryption, key creation and modification of a message. The basic conditions we are concerned about in Table 4 are that A sees all parameters prior to function application and that for the result to be fresh, one of the components must be fresh. A special example of function application is the creation of a key, as depicted in Table 5. The creation of public agreement keys requires special axioms (see [SvO94, SvO96]), and thus is treated separately here.

Table 5: Weakest preconditions for creating a key from public-agreement keys.
Operation: $A\ CreateKey : K$

Old Postcondition	Weakest Precondition	Reasoning
P $\overset{F_o(K_p,K_q)}{\longleftrightarrow}$ Q	PK$_\delta$(P,K$_p$) PK$_\delta$(Q,K$_q$)	Axiom 5

3.2 Proofs and Beliefs

Table 6: Weakest preconditions for proving a statement.
Operation: $B\ proves : \varphi$

Old Postcondition	Weakest Precondition	Reasoning
φ	ψ $\psi \supset \varphi$	Modus Ponens
φ	$\exists$ S. S controls $\varphi \wedge$ S says φ	Axiom 16

During the operation of a protocol, participants will often prove statements and develop a set of beliefs. Although these operations are typically not explicitly stated in the protocol denotation and are sometimes only implicit in the textual description of the protocol; they are essential to the operation of the protocol, and more importantly, to the validation of protocol correctness. Table 6 demonstrates the wps for proving; these are modus ponens and the axiom for control. Table 7 shows the wps for beliefs. The first is necessitation, which is only applied to theorems that can be proven from the base axioms and proof rule. The second is the application of belief to wps, where if B believes a condition after an operation, then B believes the wp of that condition prior to the operation (this permits reasoning about beliefs without duplicating each wp rule for beliefs).

Table 7: Weakest preconditions for beliefs.

Old Postcondition	Weakest Precondition	Reasoning
B believes φ	$\vdash \varphi$	Necessitation
B believes φ	B believes $wp.op.\varphi$	Wp rules

3.3 Decryption

This section presents separate weakest preconditions for decryption using each of the available key types: shared keys, public keys, private keys, and shared secrets. The notation used is:

- A $\overset{K_{ab}}{\longleftrightarrow}$ B : K_{ab} is a shared key for A and B
- $\overset{K_b}{\longmapsto}$ B : K_b is B's public key
- $\overset{K_b^{-1}}{\longmapsto}$ B : $K_{b^{-1}}$ is B's private key
- A $\overset{N_a}{\rightleftharpoons}$ B : N_a is a shared secret between A and B

Table 8: B decrypts a generically encrypted message (X_K) using a key (K).
Operation: *B decrypts* : (X_K)

Old Postcondition	Weakest Precondition	Reasoning
	* B sees $\{X\}_K$ * B sees K	
B sees X B sees X_i	True True $X_i \in X$	Axiom 12 Axiom 11
fresh(X) fresh($\{X\}_{K_b}$)	$\exists\ X_i$:$X_i \in X \land$ fresh(X_i)	Axiom 17 Axiom 18
B received X B received X_i	B received $\{X\}_K$ $X_i \in X$	Axiom 8 Axiom 7

* Necessary for B to perform decryption.

Table 8 lists the wps that are the same for all three key types. These treat decryption like a function and do not require the validity of the key. The next three tables show the unique wps for each key type: private (Table 9), public (Table 10), and shared (Table 11). These are used for determining the sender of the message and do depend upon the validity of the key (and possibly validity of a shared secret).

For messages encrypted with the public half of a public-private pair the signature depends on a valid shared secret and knowledge of who didn't send the message. Although the decrypting party doesn't have to share the shared secret in any reasonable protocol they will. For messages encrypted with the private half of a public-private pair the message is effectively signed automatically.

Messages encrypted with a shared key use the knowledge of who didn't send the message to determine the sender. This may be a difficult use of a *not* as it requires determining that the given

Table 9: B decrypts a message ($\{X\}_{K_b}$) using B's private key (K_{b-1}).

Old Postcondition	Weakest Precondition	Reasoning
A said X	C received $\{X\}_{K_b}$	Axiom 3
A sees K_b	$\xmapsto{K_b}$ B	Axiom 15
A said X_j	$\exists\ X_i{:}X_i \in X \wedge A \overset{X_i}{\rightleftharpoons} D$	Shared Secret
	$X_j \in X$	
	$\neg$ D sent $\{X\}_{K_b}$	
A says X	C received $\{X\}_{K_b}$	Axiom 3 and 19
A sees K_b	$\xmapsto{K_b}$ B	
A says X_j	$\exists\ X_i{:}X_i \in X \wedge A \overset{X_i}{\rightleftharpoons} D$	Shared Secret
	$X_j \in X$	Axiom 15
	$\exists\ X_k{:}X_k \in X \wedge$ fresh(X_k)	
	$\neg$ D sent $\{X\}_{K_b}$	

Table 10: B decrypts a message ($\{X\}_{K_{a-1}}$) using A's public key (K_a).

Old Postcondition	Weakest Precondition	Reasoning
A said X	C received $\{X\}_{K_{a-1}}$	Axiom 3
A sees K_{a-1}	$\xmapsto{K_a}$ A	Axiom 15
A said X_j	$X_j \in X$	Private Key
A says X	C received $\{X\}_{K_a}$	Axiom 3 and 19
A sees K_{a-1}	$\xmapsto{K_a}$ A	
A says X_j	$X_j \in X$	Axiom 15
	$\exists\ X_k{:}X_k \in X \wedge$ fresh(X_k)	Private Key

Table 11: B decrypts a message ($\{X\}_{K_{ac}}$) using a shared key (K_{ac}).

Old Postcondition	Weakest Precondition	Reasoning
A said X	$A \overset{K_{ac}}{\leftrightarrow} C$	Axiom 3
A sees K_{ac}	D received $\{X\}_{K_{ac}}$	
A said X_i	$X_i \in X$	Axiom 14
	$\neg$ C sent $\{X\}_{K_{ac}}$	
A says X	$A \overset{K_{ac}}{\leftrightarrow} C$	Axiom 3 and 19
A says X_j	D received $\{X\}_{K_{ac}}$	Axiom 15
A sees K_{ac}	$\exists\ X_i{:}X_i \in X \wedge$ fresh(X_i)	Axiom 3
	$X_j \in X$	
	$\neg$ C sent $\{X\}_{K_{ac}}$	

condition is not within the set of conditions. In deriving a protocol from a set of postconditions this would mean checking each operation to assure that it didn't introduce the condition.

3.4 Construction of Shared Secrets

Many protocols rely on the concept of a shared secret. This is different from a shared key in that we do not use the secret to encrypt or hide information, but rather to validate a message. In this section we present the wps for shared secrets. It is important to note that we are specifically concerned about the creation shared secrets, where we have sent the secret to another entity and await validation. Constructing a shared secret is an asymmetrical process. One of the two parties must initiate the construction by generating a fresh value to be used as a secret. The other party must finish the construction by returning the secret under construction in a context which makes it clear that no third party is being misled. In this presentation the following notation is used:

- A $\overset{N_{ab}}{\rightleftharpoons}$ B represents a valid shared secret between A and B.
- A $\overset{\underline{N_{ab}}}{\rightleftharpoons}$ B represents a shared secret for A and B which is under construction. The first of the subscripts (here 'a') denotes the initiating party.

Table 12: A completes a shared secret under construction.

Old Postcondition	Weakest Precondition	Reasoning
	* A $\overset{\underline{N_{ab}}}{\rightleftharpoons}$ B	Necessary Precondition
A $\overset{N_{ab}}{\rightleftharpoons}$ B	B said $\underline{N_{ab}}$	
A $\overset{N_{ab}}{\rightleftharpoons}$ B	A received $\{\underline{N_{ab}},B\}_{K_a}$ ¬ A sent $\{\underline{N_{ab}},B\}_{K_a}$ (A sees $K_{a^{-1}}$)	

The wps in Table 12 are used in the construction of shared secrets. Its not clear which parties can apply the operation of completing a shared secret. Clearly, belief in a completed secret can only be generated by parties who believe in the secret under construction, which will generally only be the initiating party. However, in the case that "A $\overset{\underline{N_{ab}}}{\rightleftharpoons}$ B" and "B said $\underline{N_{ab}}$" are in the set of preconditions there is no reason why Z (who may or may not have any knowledge regarding A, B, or $\underline{N_{ab}}$) couldn't apply the operation and complete the shared secret. This would not give Z any information about A, B, or N_{ab} (Z sees N_{ab} would not become true), but still seems a bit odd.

4 Proofs of Exemplary Protocols

4.1 Station-to-Station Protocol

This section presents the proofs of the first half of the station to station (STS) protocol [DvOW92]. It shows that A creates a fresh, good key (K) for use with B. The second half, showing the same for B would be quite similar.

The interesting thing about this protocol is that A produces key (K) twice. The first time A uses a function operation which allows A to see K and to believe K is fresh. The second time A uses the create key operation which proves K is a good key.

The STS protocol:

1. A $\longrightarrow$ B : R_a
2. B $\longrightarrow$ A : R_b,(B,K_b,$[B,K_b]_{K_t^{-1}}$),$\{[R_b,R_a]_{K_b^{-1}}\}_K$
3. A $\longrightarrow$ B : R_a,(A,K_a,$[A,K_a]_{K_t^{-1}}$),$\{[R_a,R_b]_{K_a^{-1}}\}_K$

Expanded protocol in our notation, from A's point of view:

STS1 A $\longrightarrow$: R_a

STS2 $\longrightarrow$ A : R_b,(B,K_b,$[B,K_b]_{K_t^{-1}}$),$\{[R_b,R_a]_{K_b^{-1}}\}_K$

STS3 A : $F(R_b,x) = K = (R_b)^x$ {A creates fresh K, but doesn't believe its good yet.}

STS4 A : decrypt $[B,K_b]_{K_t^{-1}}$

STS5 A : proves A believes $PK_\sigma(B,K_b)$

STS6 A : decrypt $\{[R_b,R_a]_{K_b^{-1}}\}_K$ {A uses K}

STS7 A : decrypt $[R_b,R_a]_{K_b^{-1}}$

STS8 A : proves A believes $PK_\delta(B,R_b)$

STS9 A : CreateKey K = $F_o(PK_\delta(B,R_b),PK_\delta(A,R_a))$ {A recreates K, believes its good}

We use the following as A's postconditions. We understand that this is not a complete set, but is sufficient to demonstrate our technique. A full set of post conditions would consist of **G1-5** as defined in [SvO96].

- A believes (A $\stackrel{K}{\longleftrightarrow}$ B) *Discharged in step 9*
- A sees $[R_b, R_a]$ *Discharged in step 7*

The generated premises, from A's point of view are:

- A sees x,R_a,K_t,$PK_\sigma(B,K_b)$
- Also A believes it sees each of the above
- A believes (fresh(x))
- A believes $PK_\delta(A,R_a)$
- A believes (T said $(B,K_b) \supset PK_\sigma(B,K_b)$
- A believes ((B said $R_b) \supset PK_\delta(B,R_b)$)

Table 13: Proof of A's postconditions for the STS protocol.

STS Step	Postconditions	Weakest Preconditions
9	A believes (A $\xleftrightarrow{K}$ B)	(8) A believes PK$_\delta$(B,R$_b$) (p) A believes PK$_\delta$(A,R$_a$)
8	A believes PK$_\delta$(B,R$_b$)	(7) A believes (B said R$_b$) (p) A believes ((B said R$_b$) $\supset$ PK$_\delta$(B,R$_b$))
7	A sees [R$_b$,R$_a$] A believes (B said R$_b$)	(p) A sees PK$_\sigma$(B,K$_b$) (6) A sees [R$_b$,R$_a$]$_{K_b^{-1}}$ (5) A believes PK$_\sigma$(B,K$_b$) (p) A believes (PK$_\sigma$(B,K$_b$)) (2) A believes (A received [R$_b$,R$_a$]$_{K_b^{-1}}$)
6	A sees [R$_b$,R$_a$]$_{K_b^{-1}}$ A believes (A sees [R$_b$,R$_a$]$_{K_b^{-1}}$) A believes (A sees $\{$[R$_b$,R$_a$]$_{K_b^{-1}}\}_K$)	(3) A sees K (2) A sees $\{$[R$_b$,R$_a$]$_{K_b^{-1}}\}_K$ (3) A believes (A sees K)
5	A believes PK$_\sigma$(B,K$_b$)	(4) A believes (T said B,K$_b$) (p) A believes (T said (B,K$_b$) $\supset$ PK$_\sigma$(B,K$_b$))
4	A sees K$_b$ A believes (T said B,K$_b$)	(p) A sees K$_t$ (2) A sees [B,K$_b$]$_{K_t^{-1}}$ (2) A believes (A received [B,K$_b$]$_{K_t^{-1}}$) (p) A believes PK$_\sigma$(T,K$_t$)
3	A sees K A believes (A sees K) A believes (fresh(K))	(2) A sees R$_b$ (p) A sees x (2) A believes (A sees R$_b$) (p) A believes (A sees x) (p) A believes (fresh(x))
2	A sees R$_b$ A sees (B,K$_b$,[B,K$_b$]$_{K_t^{-1}}$) A sees $\{$[R$_b$,R$_a$]$_{K_b^{-1}}\}_K$ A sees [B,K$_b$]$_{K_t^{-1}}$ A believes (A sees R$_b$) A believes (A received [B,K$_b$]$_{K_t^{-1}}$) A believes (A received [R$_b$,R$_a$]$_{K_b^{-1}}$)	
1	A sent R$_a$	(p) A sees R$_a$

These premises are generated through direct application of the wp rules given in the previous section. Since we are generating preconditions from postconditions, we work backwards from the last step of the protocol to the first. In Table 13 we have labeled the generated preconditions with either the step in which they are discharged, or with a (p) to denote that they should be considered a premise of the protocol. For example in step 9 we apply the CreateKey rule from Table 5 to the valid key creation postcondition, the generated preconditions are discharged in step 8 and as a premise. In step 8 we apply the proof wps from Table 6 to the wp generated in step 9, the generated preconditions are discharged in step 8 and as a premise.We continue with this through the remainder of the protocol.

4.2 Needham-Schroeder key distribution protocol

This section presentes the use of the weakest precondition calculus to prove the Needham–Schroeder protocol for key distribution (not for public keys). The original protocol is:

1. A $\longrightarrow$ S: A,B,N_a
2. S $\longrightarrow$ A: $\{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{bs}}\}_{K_{as}}$
3. A $\longrightarrow$ B: $\{K_{ab}, A\}_{K_{bs}}$
4. B $\longrightarrow$ A: $\{N_b\}_{K_{ab}}$
5. A $\longrightarrow$ B: $\{N_b\text{-}1\}_{K_{ab}}$

The expanded protocol from A's point of view is:

NS-A1 A $\longrightarrow$: A,B,N_a {A sends a message}

NS-A2 $\longrightarrow$ A: $\{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{bs}}\}_{K_{as}}$ {A receives the message}

NS-A3 A: decrypt$\{X\}_{K_{as}}$ {A decrypts the message}

NS-A4 A proves (A $\overset{K_{ab}}{\longleftrightarrow}$ B)

NS-A5 A proves (fresh(K_{ab})

NS-A6 A $\longrightarrow$ $\{K_{ab}, A\}_{K_{bs}}$

NS-A7 $\longrightarrow$ A: $\{N_b\}_{K_{ab}}$

NS-A8 A: decrypt$\{N_b\}_{K_{ab}}$

NS-A9 A: F(N_b,1) = N_b-1

NS-A10 A: F(N_b-1,K_{ab}) = $\{N_b\text{-}1\}_{K_{ab}}$

NS-A11 A $\longrightarrow$ $\{N_b\text{-}1\}_{K_{ab}}$

Steps NS-A1 through NS-A5 are used in testing A's postconditions. The postconditions for A should be:

- A believes (A sees K_{ab}) *Discharged in step NS-A2*

- A believes (A $\stackrel{K_{ab}}{\longleftrightarrow}$ B) *Discharged in step NS-A4*
- A believes (fresh(K_{ab})) *Discharged in step NS-A5*

Note that we do not include a post-condition that A sent any message to initiate the protocol. We strongly believe that such a post condition is important, and that specific ordering between protocol steps should be maintained. This is discussed further in [AF97].

Table 14: Proof of A's postconditions for NS protocol.

NS-A Step	Postconditions	Preconditions
5	A believes (fresh(K_{ab}))	(3) A believes (S says K_{ab}) (p) A believes (S controls fresh(K_{ab}))
4	A believes (A $\stackrel{K_{ab}}{\longleftrightarrow}$ B)	(3) A believes (S says K_{ab}) (p) A believes (S controls A $\stackrel{K_{ab}}{\longleftrightarrow}$ B)
3	A believes (S says K_{ab}) A sees $\{\{K_{ab},A\}_{K_{bs}}\}_{K_{as}}$	(p) A believes (A sees K_{as}) (p) A believes (A $\stackrel{K_{as}}{\longleftrightarrow}$ S) (p) A believes (fresh(N_a)) (2) A believes (A received $\{X\}_{K_{as}}$) (p) A sees K_{as} (2) A received $\{X\}_{K_{as}}$ X = $\{N_a,B,K_{ab},\{K_{ab},A\}_{K_{bs}}\}$
2	A believes (A received $\{X\}_{K_{as}}$) X = $\{N_a,B,K_{ab},\{K_{ab},A\}_{K_{bs}}\}$ A believes (A sees K_{ab})	
1	A sent (A,B,N_a)	(p) A sees "A" (p) A sees "B" (p) A sees N_a

The proof of A's postconditions (steps 1 through 9 of the protocol) are presented in Table 14, generating the following preconditions for A:

- A believes fresh(N_a)
- A believes (S controls A $\stackrel{K_{ab}}{\longleftrightarrow}$ B)
- A believes (S controls fresh(K_{ab})
- A believes (A $\stackrel{K_{as}}{\longleftrightarrow}$ S)
- A sees "A", "B", N_a, and K_{as}
- A believes (A sees K_{as})

It's now been proven that A believes (A $\stackrel{K_{ab}}{\longleftrightarrow}$ B) and that A believes (fresh(K_{ab})). However, steps 5 and 4 are not entirely correct. In step 8 "S says K_{ab}" is treated as "S says A $\stackrel{K_{ab}}{\longleftrightarrow}$ B". Either "A believes (S says K_{ab} $\supset$ S says A $\stackrel{K_{ab}}{\longleftrightarrow}$ B)" should be a premise or "K_{ab}" should be idealized as "A $\stackrel{K_{ab}}{\longleftrightarrow}$ B" in S's message to A. Similarly in step 9 "A believes (S says K_{ab})" is used where "A believes

(S says (fresh(K_{ab})))" is necessary. Again the two solutions are a premise saying "A believes (S says $K_{ab} \supset$ S says fresh(K_{ab})" or an idealization. For both of these problems Syverson and van Oorschot [Syv96] use the premise approach, we have opted for protocol idealization.

The expanded protocol from B's point of view is:

NS-B1 $\longrightarrow$ B: $\{K_{ab},A\}_{K_{bs}}$

NS-B2 B: decrypt$\{K_{ab},A\}_{K_{bs}}$

NS-B3 B proves (A $\xleftrightarrow{K_{ab}}$ B)

NS-B4 B proves fresh (K_{ab})

NS-B5 B: F(N_b,K_{ab}) = $\{N_b\}_{K_{ab}}$

NS-B6 B $\longrightarrow$ $\{N_b\}_{K_{ab}}$

NS-B7 $\longrightarrow$ B: $\{N_b\text{-}1\}_{K_{ab}}$

NS-B8 B: decrypt $\{N_b\text{-}1\}_{K_{ab}}$

Steps NS-A1 through NS-A5 are used in testing A's postconditions. The postconditions for B should be:

- B believes (B sees K_{ab}) *Discharged in step NS-B2*
- B believes (A $\xleftrightarrow{K_{ab}}$ B) *Discharged in step NS-B3*
- B believes (fresh(K_{ab})) *Discharged in step NS-B4*
- B believes (B received $N_b - 1$) *Discharged in step 8*

The first three postconditions are symmetric to those of A. The last is used to help prove currency, and requires an additional premise that N_b is fresh. One difficulty in this proof comes from conditions of the form $\neg$ B sent $\{K_{ab},A\}_{K_{bs}}$. The truth of this negated conditions comes from making sure that B sent $\{K_{ab},A\}_{K_{bs}}$ isn't in any of the preconditions. This may raise complications that aren't there for the other conditions which can be proven in a single step.

The proof of B's postconditions is shown in Table 15 and generates the following preconditions for B:

- B believes fresh(N_b)
- B believes (S controls A $\xleftrightarrow{K_{ab}}$ B)
- B believes (S controls fresh(K_{ab})
- B believes (B $\xleftrightarrow{K_{bs}}$ S)
- B sees K_{bs},N_b
- B believes (B sees K_{bs})

Table 15: Proof of B's postconditions for NS protocol.

NS-B Step	Postcondition	Preconditions
8	B believes (B received N_b-1)	(7) B believes (B received $\{N_b\text{-}1\}_{K_{ab}}$) (7) B believes (B sees $\{N_b\text{-}1\}_{K_{ab}}$) (2) B believes (B sees K_{ab})
7	B believes (B received $\{N_b\text{-}1\}_{K_{ab}}$) B believes (B sees $\{N_b\text{-}1\}_{K_{ab}}$)	
6	B sent $\{N_b\}_{K_{ab}}$	(5) B sees $\{N_b\}_{K_{ab}}$
5	B sees $\{N_b\}_{K_{ab}}$	(p) B sees N_b (2) B sees K_{ab}
4	B believes fresh (K_{ab})	(p) B believes (S controls fresh (K_{ab})) (2) B believes (S says fresh (K_{ab}))
3	B believes A $\xleftrightarrow{K_{ab}}$ B	(p) B believes (S controls (A $\xleftrightarrow{K_{ab}}$ B)) (2) B believes (S says A $\xleftrightarrow{K_{ab}}$ B)
2	B believes (S said A $\xleftrightarrow{K_{ab}}$ B) B believes (B sees K_{ab})	(p) B believes (B sees K_{bs}) (p) B believes (B $\xleftrightarrow{K_{bs}}$ S) (?) $\neg$ B sent $\{K_{ab}, A\}_{K_{bs}}$ (1) B believes (B received $\{K_{ab}, A\}_{K_{bs}}$)
1	B believes (B received $\{K_{ab}, A\}_{K_{bs}}$)	(10) $\exists$ C:C sent $\{K_{ab}, A\}_{K_{bs}}$)

The problems arise in steps 2, 3, and 4. First, just as for participant A, the server S only mentions the key K_{ab} and does not explicitly mention its freshness or validity. Unlike in A's proof an idealization (e.g. K_{ab} idealized as fresh (K_{ab})) won't work for B. Because of the nonce N_a participant A could believe "S says K_{ab}", but at best B can only prove "S said K_{ab}" (step 12). Thus, the proof:

S says fresh (K_{ab}) $\wedge$ S controls fresh (K_{ab}) $\supset$ fresh (K_{ab})

doesn't work. Instead a premise such as:

S said fresh (K_{ab}) $\supset$ fresh (K_{ab})

is necessary. Of course, using said in the place of says is quite risky as B can be subjected to a replay attack using old, broken keys for K_{as} and K_{ab} even if the key K_{bs} remains valid. Again, we have assumed a protocol idalization to remove this problem. Further work is definitely needed to resolve this problem in the verification of this protocol.

5 Discussion

In this paper we presented a weakest precondition calculus for the verification of authentication of authentication protocols. The benefits of this approach to protocol verification include automatic generation of protocol preconditions, analysis of only recipient beliefs, and detailed specification of the protocol steps. In this section we present some of our thoughts on the benefits and drawbacks of this approach. We then present a brief discussion outlining an approach to formal protocol derivation as a derivative of this work.

5.1 The Weakest Precondition Calculus

Two of the goals of using weakest preconditions are to verify that authentication protocols meet their stated postconditions (as was done with the STS and the Needham-Schroeder protocols) and possibly to write authentication protocols which meet a desired set of postconditions. It is also hoped that both of these tasks can be fully or, at least, partially automated.

The advantages of the weakest preconditions for protocol verification are that they seem to require a minimum of premises (in contrast to the SvO logic) and a minimum of idealizations (in contrast to the BAN logics). The more premises and idealizations that must be included the more likely it is that one of them will be incorrect or assume to much and thereby hide a flaw in the protocol.

However, several features of the current weakest preconditions complicate the verification, and possibly the writing, of authentication protocols. The use of a *not* for several of the weakest preconditions means that these preconditions cannot be met by a single operation, but rather every operation must be examined to assure that it doesn't introduce a condition which would violate the *not*.

In many cases the same postcondition can be met by several different operations, which often require different preconditions. (For example, different encryption schemes means there are several ways of proving who sent a message.) This will make automatic generation of protocols more difficult because there will be several, often nearly equivalent, methods of meeting the desired postconditions leading to many potential protocols. One partial solution is to restrict the options in advance. For example, only allowing one type of key to be used.

Another important task should be the verification of the weakest preconditions themselves. Although based on the SvO logic which is sound, the mixture of axioms and semantics used in defining the weakest preconditions makes them suspect. Additionally, they incorporate several features (such as constructing shared secrets) which are not clearly presented in the SvO logic.

5.2 Protocol Derivation

After reviewing the protocol analysis provided in Section 4 there is an apparent strategy to the proofs. This strategy involves specification of the protocol in expanded form, from the point of view of the recipient; generation of the set of post conditions for the protocol; and then systematic generation of the set of preconditions for the protocol. What if we took this one step further? Suppose that we only defined the post conditions of the protocol and used a variable, Pl to denote the list of protocol steps. Define Pl as $Pl_1 : P_a$, which means Pl is a list of steps that terminates with step P_a. Choosing an appropriate P_a, we can generate the wps for P_a with respect to the given postconditions. We can now use these as postconditions for Pl_1. Following this process we can incrementally unwind the protocol specification generating preconditions as we go. Once we have reached a set of satisfactory preconditions, we can stop.

This is not trivial, and definitely not readily automatable, but there we can develop some heuristics to assist in this effort. For example, we can search the list of postconditions for those which correspond to an appropriate weakest precondition rule. We then insert the operation associated with the rule and modify the preconditions. Any precondition of a form that is typical of protocol premises (such as A sees its own public and private keys) can be ignored and propagated up.

It is our belief that with some work, now underway at the University of Idaho, we can develop a set of heuristics that will enable us to formalize the derivation of protocols.

References

[AF97] J. Alves-Foss. The use of belief logics in the presence of causal consistency attacks. In *Proc. National Information Systems Security Conference*, Oct. 1997.

[BAN89] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. Technical Report 39, DEC Systems Research Center, Palo Alto, CA, February 1989.

[DvOW92] W. Diffie, P. van Oorschot, and M. Wiener. Authentication and authenticated key exchanges. *Design Codes and Cryptograhpy*, 2:107–125, 1992.

[GNY90] L. Gong, R. Needham, and R. Yahalom. Reasoning about belief in cryptographic protocols. In *Proc. IEEE Symposium on Research in Security and Privacy*, pages 234–248, 1990.

[Mea96] C Meadows. Analyzing the Needham-Schroeder public key protocol: A comparison of two approaches. In *Proc. ESORICS 96*. Springer Verlag, 1996.

[NS78] R.M. Needham and M.D. Schroder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[Sne91] E. Snekkenes. Exploring the BAN approach to protocol analysis. In *Proc. IEEE Symposium on Research in Security and Privacy*, pages 171–181, 1991.

[SvO94] P. Syverson and P. van Oorschot. On unifying some cryptographic protocol logics. In *Proc. IEEE Symposium on Research in Security and Privacy*, pages 14–28, 1994.

[SvO96] P. Syverson and P. van Oorschot. A unified cryptographic protocol logic. Unpublished manuscript, Dec. 1996.

[Syv93] P. Syverson. Adding time to a logic of authentication. In *Proc. First ACM Conference on Computer and Communications Security*, pages 97–101. ACM Press, 1993.

[Syv96] P. Syverson. A new look at an old protocol. *Operating Systems Review*, 30(3):1–4, July 1996.