Cryptographic Protocols: An Overview of Concepts and Methods

Nancy N. Mogire
Department of Information and Computer Sciences
University of Hawaii at Manoa
Honolulu, USA
nmogire@hawaii.edu

Abstract

Cryptographic protocols provide the framework for different internet technologies to interact so that messages can be sent and received in the intended form in a timely manner and without any tampering by intruders along the way. Since messages are transported in packets across multiple network components, the risk of inteference is imminent. Network components can be reprogrammed by intruders who can then intercept messages and either prevent them from reaching the intended receiver or tamper with and relay them in an altered form. The goal of protocol security is to ensure that protocols are designed in a way that they can resist interference. There is need for security of a protocol be provable in order for systems implementing it to be reasonably expected to run securely. It is around this need that the science of protocol security has emerged. This paper provides an overview of the background and reasoning behind the design of cryptographic protocols, and later provides an overview of the development of methods for formalization of protocol description and verification. Finally, this document presents an application of diagrammatic description of protocols and an application of protocol proving axioms from protocol derivation logic in the verification two protocols CRE and CREE-seq.

Index Terms

Actor Networks; Protocol Verification; Protocol State Machine; Cryptographic Protocols; Protocol Proving Axioms; Protocol Process; Adversary; Intruder; Protocol Taxonomy; Protocol Correctness; Key Exchange; Key Establishment; Formal Methods; Security properties; Protocol participants; Session Key; Session;

I. Introduction

A. Achievements and outline of the report

This report achieves the goal of providing a brief overview of the science of cryptographic protocol design and analysis. It also presents a study on the methods of formalization of protocol descriptions and verification. The remainder of this document is organized as follows; In section II we present a literature review on protocols. Specifically, in part A we delve into a description of how protocols work, their goals and various classification systems; Part B is a detailed overview of cryptographic protocols in general and of some selected network implementations; In section 3A we discuss the development of methods to formalize protocol description and verification. In part B we present an application of diagrammatic description of protocols and an application of protocol proving axioms from protocol derivation logic in the verification two protocols CRE and CREE-seq. Section IV is a conclusion on the work in this document.

II. Literature Review

A. Background

A very general definition of a protocol is ”a system of rules that explain the correct conduct and procedures to be followed in formal situations” (Merriam Webster Online, 2015). In the context of computing, protocols are primarily used to define the interactions of multiple parties to establish connections and exchange messages. There are several types of protocols depending on the network layer at which each one occurs. Two examples for protocols at the application layer are the hypertext transfer protocol(HTTP) and the simple mail transfer protocol(SMTP).

Each protocol has its own formal definition. Every protocol is defined within the context of some communication architecture. In this document we focus on a generic scenario where two client computers which we call A and B, want to be able to exchange messages securely over a cyberchannel. Since messages are transported in packets across various network components and locations, they are susceptible to intrusion. Intruders can reprogram network components at convenient points for them to intercept messages and either prevent them from reaching the intended receiver or to alter them and relay them in an altered version. Hence the cyberchannel is said to be insecure.

The problem A and B face is the need to establish a shared session key to enable them exchange encrypted messages and hence to secure their communication over the network during the session. The two principals A and B can either make use of a trusted server to randomly generate and distribute a session key to them or they could use public key cryptography to each privately generate a shared key. The purpose of separating communication into sessions is to restrict access to each message run to a specific set of principals and to a reasonable timeframe. It is assumed that an attacker eventually learns the key given a sufficient amount of time. The important feature of a session key is that it should be unpredictable by the adversary.

In protocols security, we assume perfect cryptography and abstract away from the details of cryptographic implementations. Instead we focus on the primary task of a protocol which is to make the key K^{AB} known to A and B and no other party except possibly the trusted server S. The idea then is to authenticate A and B so they receive information about the key. The authentication problem arises from the fact that the adversary controls the communication channel such that they are able to alter, insert, delete or re-route messages in the protocol. The extent of such ability varies based on various properties of the networks and as such the specific impact is not always obvious.

Beyond making the key available to the principals there are certain other properties that a protocol should guarantee in order for such key to be expected to provide security for the session. For instance the protocol should assure both A and B that K^{AB} is newly generated, which is known as the freshness property and it ensures that a principal does not receive a replay of an old key that an adversary may already know. Protocols should also assure confidentiality and integrity. Every protocol is designed with some specific set of primary goals based on what type of security is desired for the specific communication. These goals are given a more detailed treatment later in this section. As it will become apparent, the goals of a protocol determine whether the protocol is considered secure or not based on whether these goals are achieved.

So far, the tradition in protocol design is to specify the structure of a successful protocol run i.e. the expected message sequence. As such, no error handling is envisioned and protocols do not provide for message validation. Additionally, since the actions of principals are not detailed in the protocol it is not obvious how principals deal with decision making in case several options are available e.g. of different message constructions. However, there is an established practice around protocol analysis that is intended to prove protocols secure. While there has been work on writing protocol proofs, a big part of protocol analysis is attempting to design attacks on protocols and for as long as an attack is not found on a given protocol it is assumed to be secure.

Protocols can be classified in various ways but three major criteria are; whether or not shared keys are already established; How the session key is generated and how many users the protocol is designed to serve.

With regard to the first criterion, there are three cases. The first case assumes that the principals who want to generate a session key already share a secret key. The second case assumes that the principals have certified public keys. The last case assumes that an online server who is a trusted third party mediates the key exchange. The second mode of classification also distinguishes three groups of protocols. The first group consists of key transport protocols where one of the principals generates the key to be transported to all other users of the protocol. The second second group consists of key agreement protocols where the session key is a function of inputs by all protocol users. The third type of protocols are hybrids being key transport protocols for some users and key agreement protocols for

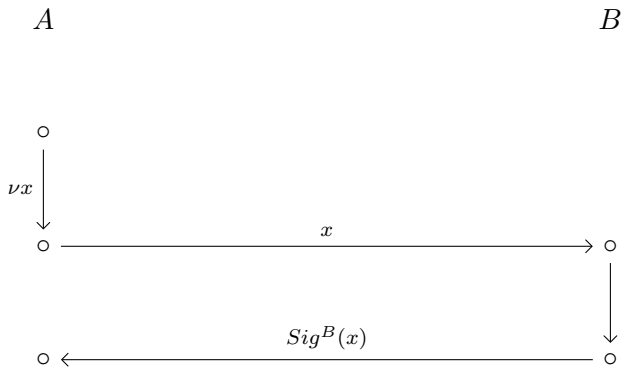
others. The idea is that the session key is a function of inputs from more than one but not all users in the protocol. The subset of users that participates in generating a key are the ones refered to as principals. With respect to classification by number of users, protocols can be point to point serving only two users in any given run or they can be conference key protocols that will have multiple participants.

B. Protocol Goals

There are a number of protocol design goals in any communication architecture. Here we review some of the most important goals. Boyd et al[1] categorize these goals into the primary goals which are the user-oriented goals and the key-oriented goals. There are other goals which fall under a third category that he calls the enhanced goals and these serve the function of strengthening the primary goals. Here we discuss the two primary goals.

The first goal is *key establishment*. This is where the protocol is intended to establish a fresh session key known only to the participants in the session. Menezes et al[5] defines key establishment as the process by which a shared key becomes available to two or more parties for subsequent cryptographic use. Successful key establishment leads to a good session key. Boyd et al[1] describe the properties of a good session key. To begin with, from principal A's perspective a shared session key is good to use with B if A is assured of the Key's freshness and the key is only known to A, B and any trusted third parties. B determines if the session key is good by similar reasoning. A public session key is good if freshness is guaranteed and the private key is only known to the owner. A key is good if it guarantees integrity i.e. all inputs for its generation come from legitimate participants and if it cannot be modified by the adversary.

Another goal of protocols is *entity authentication*. Menezes et al[5] provide a definition for entity authentication as follows; "For one party to acquire corroborative evidence of identity of another party involved in the protocol and that such party has actually participated". Authentication can be achieved in various ways including requiring the party to be authenticated to provide a signature on some message as in the example below:



Boyd' et al[1] break down entity authentication more specifically; a principal A is said to have knowledge of B as their peer in a protocol if A is aware of B as their claimed peer entity in the protocol. Strong entity authentication of A to B can be claimed for a protocol if B has fresh assurance that A has knowledge of B as its peer and A is operative. Mutual authentication is achieved if both entities authenticate each other in the same protocol, otherwise authentication is unilateral. Perhaps the reason Boyd et al categorize the remainder of the protocol goals as enhanced is because they arise in building up towards either one of the two first goals. These other goals include Key confirmation, key freshness and mutual understanding of a shared key i.e. both A and B have the same key k and each is certain that the other knows key k.

C. Important Properties of Protocols and the Role of Cryptography

Expressing properties of a good protocol is indeed another way of formulating how the goals of the protocol can be achieved. Here we discuss some properties used in proving the security of protocols. A good protocol requires to have the property that it *guarantees the origin of a message*. The origin of a message can only be guaranteed if the message has not changed since it was originated. Use of encryption, digital signatures and hash functions aids in ensuring integrity. Pavlovic[6] presents two protocol proving axioms arising from this property:
The first is the origination axiom (rcv) that says that if some message t is received by principal A then there must be another entity X such that X originated and sent t and subsequently it was received by A:

$((t))_A \implies \exists X. << t >> \underset{X}{\rightarrow} > ((t))_A$ (rcv)
and the encryption axiom (enc) says that if something is encrypted under some principal X's cryptographic key, if it is later decrypted then it must have been decrypted by that principal X.

$A : (\nu x)_A > << E^B(t(x) >> \underset{A}{\rightarrow} > << x >> \underset{x}{\rightarrow}$
$\implies X = A \vee X = B$ (enc)

Another important property of protocols is that the *freshness of key* should be guaranteed for any given session key generated. There is a whole class of protocol attacks that is based on the adversary gaining knowledge of a key from an old session and replaying it to some principal while impersonating the user with whom it was previously shared and creating a false new session with the unsuspecting principal. To guarantee security, it is important that each user is able to verify that a key gained by the end of the protocol is new and not a replay from a previous session. For this to be guaranteed it is important that a principal have a part in choosing the key value. The user can hence incorporate or otherwise check for a value that they know to be fresh. Such value may be a nonce i.e. a number generated for one-time use, or the value can be some a timestamp or a counter value.

For instance if the session key for A and B is a function of their inputs i.e. $K^{AB} = f(N^A, N^B)$ then from A's view once she chooses a freshness value N^A, B cannot choose N^B such that K^{AB} is an old value. The freshness value is often encrypted and sent as a challenge and a correct response provides entity authentication. Pavlovic[6] formalizes the freshness proving axiom as follows:

$(\nu x)_A \wedge x \in Fv(a_B) \implies ((\nu x)_A > (a_B)) \wedge$
$A \neq B \implies (\nu x)_A > << x >>_A > ((x))_B > a_B$ (new)

An additional property of protocols that is important but not directly used to prove security is efficiency. A protocol need to be computationally efficient such that an algorithm can be developed to implement it reasonably. As such one way to optimize a protocol is to minimize the number of message rounds involved in a given run. Where a round refers to all messages that can be sent in parallel from one side to the other. There are other properties that enhance the security of a protocol including forward secrecy and key compromize impersonation resistance.

D. Attacks on Protocols

There are several classes of attacks on protocols. Boyd et al[1] discuss several of them in detail. An example is the class known as replay attacks. Replay attacks comprise of the adversary tampering with a protocol run by inserting a some message from an earlier session or otherwise a message that has been sent earlier in a parallel session. The adversary's goal here is to try and cause the protocol run to result in a session key to which they are privy. Another example is a modification attack which involves the adversary altering messages running between two principals or sometimes splitting and reorganizing messages and thereby tampering with their integrity.

III. Formal Methods of protocol description and verification

Formalizing protocol description and verification seeks to address the problem that informal arguments for protocol correctness are not reliable. Such informal arguments often lead to loopholes going unnoticed in protocols and being exploited later by attackers. Generally, there are two types of formal methods for protocol analysis; Model checking and theorem proving.

Model checking considers a large but finite number of protocol behaviors. Model checkers enable checking against a set of correctness conditions. This approach is good for finding attacks rather than proving correctness and it is easier to automate. Model checkers easily provide a counterexample for incorrect protocols but are uninformative when a protocol is correct as they do not provide a reason. An example of a model checker is FDR developed by Lowe[4]. This model checker can check the process algebra CSP in which communication is modelled by the notion of channels. Each principal is modelled as a CSP process together with its steps in the protocol. The model checker checks for refinement between CSP processes and the property in question. If this refinement does not occur then an attack can be constructed on the protocol. The limitation of scope for this model checker was solved when Lowe proved mathematically that an attack on the smaller protocol implies an attack on the general protocol. Other model checkers include *Mur*Φ and *Brutus* which was designed specifically for cryptographic protocols.

NRL Protocol Analyzer lies between model checking and theorem proving. This analyzer specifies the system state, protocol rules and rewrite rules. In NRL analysis begins from an insecure state, checking to see if that state can be reached from the initial state of a system. If an insecure state is reachable using the given protocol, then an explicit attack can be constructed for it.

Theorem proving considers all possible protocol behaviors and allows checking against expected protocol behaviors. This system is suited for proving correctness of protocol rather than for finding attacks and it is less automated. Theorem proving provides symbolic reasoning for claiming that a protocol is correct. Their limitation is that they are largely manual and need a lot of interaction from an expert user. An example of a theorem proving system is BAN logic. In BAN logic the principals, keys and nonces form the primitive working objects. Messages are expressed as a formula of the logic. If an unreasonable assumption is found in a protocol process then it suggests a protocol flaw. Another system is the *strand spaces model*. This framework provides for more mathematical rigour. In this system sent and received messages including those of the adversary are represented as strands. Elements of a strand are called nodes and the set of all possible strands is the strand space. Properties are proved by induction on well-founded sets.

The formalization work and proofs later in this document pick up from the framework known as *protocol derivation* logic due to Cervesato, Meadows and Pavlovic[2]. In this framework, protocol fragments are composed and patterns matched to properties. Each protocol participant in this case observes some local events and makes some deductions about the peers in the protocol. The system enumerates properties supported by the protocol based on its construction. When an expected property does not manifest, this indicates a missing component or component failure, and a counterexample can be constructed. In this framework there are protocol taxonomies to help in understanding, choosing and devising protocols based on desired features.

IV. Formalized description and analysis of two selected protocols

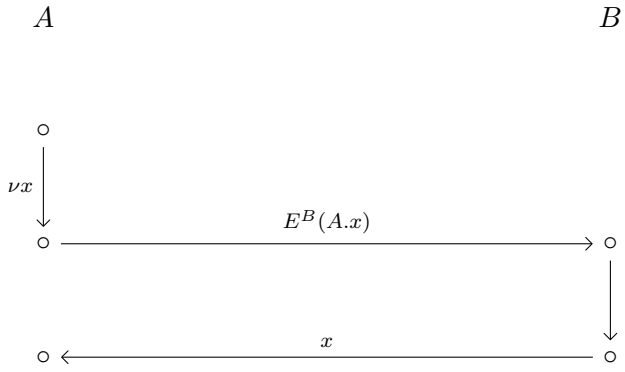
A. Introduction

In this section we adopt the approach of protocol description by diagrammatic representation of the movement of messages within a protocol run as observed by the author in the work of Cervesato, Meadows and Pavlovic[2]. We then proceed to apply the axioms of theorem proving stated in section II,C. to analyze two implementations of challenge response authentication in CRE and CREE-Seq. The first employs encryption only from the initiating

client while the second employs encryption in both directions, and authentication is sequential such that the initiating client authenticates the peer first before being authenticated by the peer. We check if the two protocols implement strong authentication. Strong authentication means that the participants realize matching conversations by the end of the protocol and that they also achieve mutual authentication. In these analyses, honesty of participants is assumed and it is also assumed that cryptographic keys are uncompromized.

B. CRE

This protocol is formally expressed as $(CR)[C^{AB}x = E^B(A.x), r^{AB}x = x]$ indicating that the challenge is encrypted by A using B's public key and the reponse is the decrypted challenge sent back to A. Diagrammatically, this is represented as below:



In this protocol Alice generates a nonce and sends it to B with her identity bound in it. The nonce is not required to be random but it should be unpredicatable to the adversary. Subsequently she receives x from a node in the network. This implies that someone must have sent it since by the origination axiom, if something is received it must have been sent i.e.

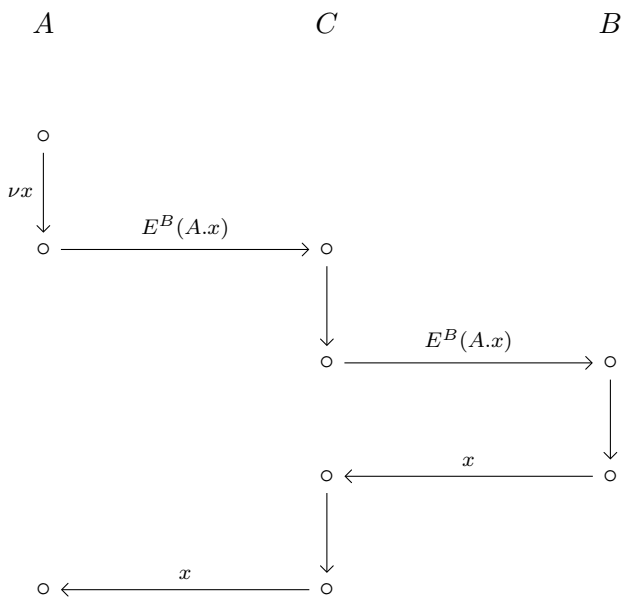
$((x))_A \implies \exists X. << x >> \underset{X}{\rightarrow} > ((x))_A$ (rcv)
This statement says that if A received a message containing t then there must be some X such that X originated a message containing t which was subsequently received by A. A determines that the received x is new since she just recently generated it and sent is out. By the freshness axiom we know that:

$(\nu x)_A \wedge x \in Fv(a_B) \implies ((\nu x)_A > (a_B)) \wedge$
$A \neq B \implies (\nu x)_A >< << x >>_A > ((x))_B > a_B$ (new)

The fact that x was received and decrypted by B at some point is determined from the fact that only B could have had the correct private key to decrypt the message since it was encryped using his public key. By the encryption axiom (enc) we know that:

$A : (\nu x)_A >< << E^B(t(x) >> \underset{A}{\rightarrow} >< << x >> \underset{x}{\rightarrow}$
$\implies X = A \vee X = B$ (enc)
Since x is received from an external source, $X \neq A$ hence $X = B$.

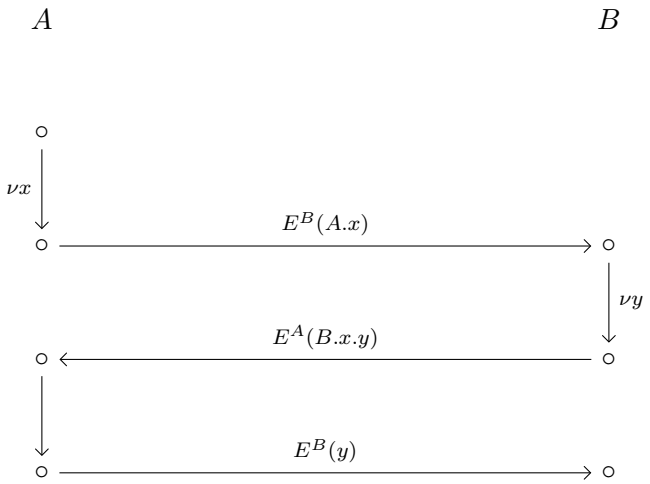
However there is no guarantee that x on its way from B was not intercepted by a man-in-the-middle who later relayed in back to A. the possible attack is illustrated below:



hence this protocol authenticates A to B and assures to A that B was recently alive, but nothing more.

C. CREE-Seq

This protocol is illustrated below:

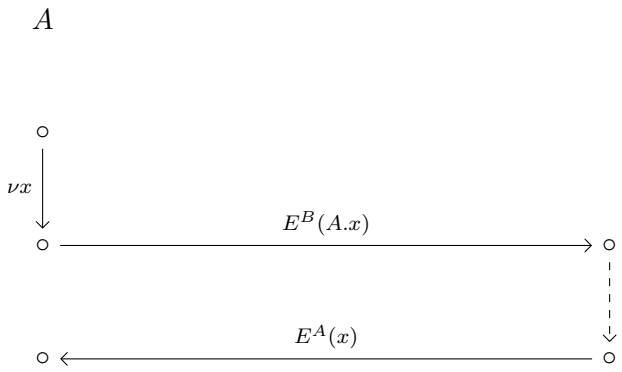


We claim that this protocol realizes strong authentication. i.e. it achieves matching conversations for both parties and provides mutual entity authentication.

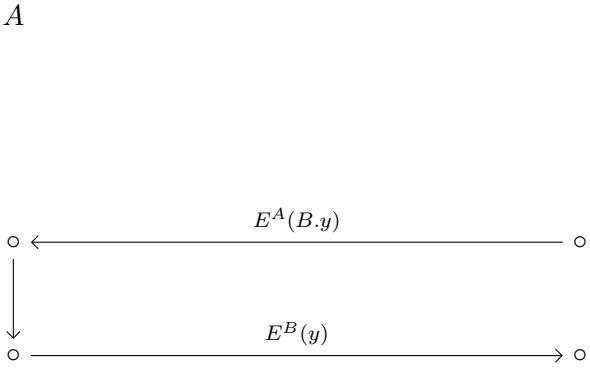
1) matching conversations: Alice ś analysis:
Alice sends a nonce x and subsequently receives a message containing x. Freshness of x can be proved by the freshness axiom (new):

$(\nu x)_A \wedge x \in Fv(a_B) \implies ((\nu x)_A > (a_B)) \wedge$
$A \neq B \implies (\nu x)_A ><< x >>_A> ((x))_B > a_B$ (new)

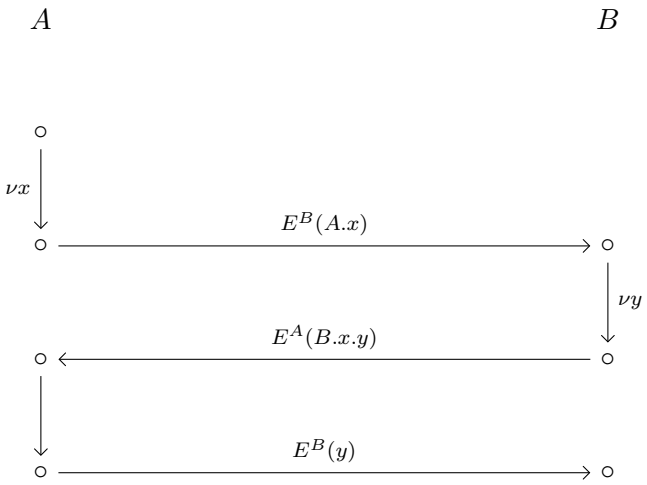
Her CR view is as below:



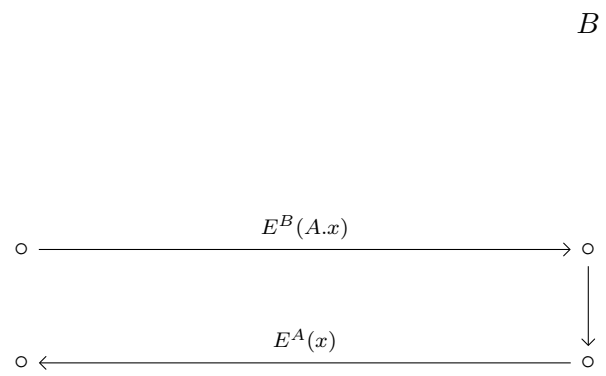
But Alice knows that she also received a nonce y bound to B ś identity and sent back y encrypted with B's PKE:



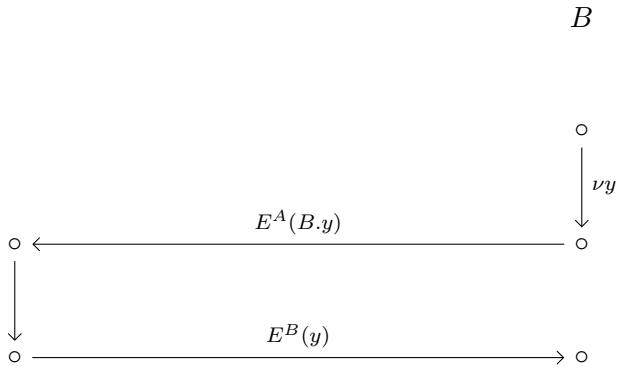
Composing the two she derives:



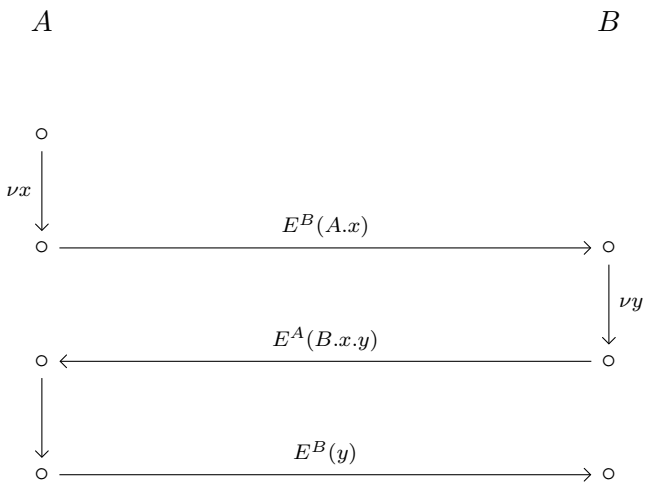
Similarly, Bob initially knows:



and



composing the two he derives:



Hence matching conversations goal is realized.

2) mutual authentication: Mutual entity authentication is realized as follows:
Here Alice knows that she has generated a nonce x and sent it encrypted under Bob's public key and with her identity bound to the message. Subsequently she receives x encrypted under her own public key and with B's identity bound to the message as well as a new nonce not generated by herself. This implies by the receive axiom (rcv) that someone else must have sent this message:
$((x))_A \implies \exists X. << x >>\underset{X}{\rightarrow}> ((x))_A$ (rcv)

The sender must be B since the initial message containing A's challenge was sent out under B's public key and we know by the encryption axiom (enc) that:
$A : (\nu x)_A >< E^B(t(x) >> \underset{A}{\rightarrow} >< x >> \underset{x}{\rightarrow}$
$\implies X = A \vee X = B$ (enc)
Since x is received from an external source, $X \neq A$ hence $X = B$.

Additionally, B's identity is bound to the message cryptographically and we can assume that the message arrives directly from B since if A's public key is not compromized an attacker would not be able to decrypt the message and modify it logically before relaying it. Hence challenge-response authentication of B is achieved for A. B also achieves CR authentication of A by similar reasoning since they also send an encrypted nonce to A and subsequently receive it back encrypted.

Since the protocol achieves both mutual authentication and matching conversations, it achieves strong authentication.

V. Conclusion and Future work

In this work we apply some axioms in the derivational system of protocol proving to analyse protocols. The task we see is to develop an even simpler language for protocol derivation as well as description to cover an emerging class of more complex protocols more susceptible to ambiguity and error. In this document we restrict ourselves to the scenario of a protocol between two clients and the potential involvement of a trusted server as a third party. However, the real challenge in protocol security research is the analysis of protocols in a more complex setup that includes not just clients interacting across cyberspace but also physical objects which may themselves be changing form and location in the course of a protocol run. It gets more complicated since there is no global observer of the cyberphysical network and instead each participant in the protocol only makes local observations and needs to reach some security deductions to inform his actions in the protocol. This kind of situation needs a more versatile framework for deriving expected protocol behaviors and as such proving protocols correct.

References

[1] C. Boyd, A. Mathuria, *Protocols for Authentication and Key Establishement*, ISBN: 3-540-43107-1, Springer, 2003.
[2] I. Cervesato, C. Meadows, D. Pavlovic, *Deriving Key Distribution protocols and their security properties*, 2006.
[3] F. Fabrega, *Strand spaces: proving security protocols correct*, Volume 7 Issue 2-3, Journal of Computer Security, Pages 191 - 230, Jan. 1999.
[4] G. Lowe, *Breaking and fixing the Needham-Schroeder public key protocol using FDR.* In Tools and Algorithms for the Construction and Analysis of Systems, pages 147-166. Springer-Verlag, 1996.
[5] A. Menezes, M. Qu, and S. Vanstone. *Some new key agreement protocols providing implicit authentication.* In Workshop on Selected Areas in Cryptography (SAC'95), pages 22-32, 1995.
[6] D. Pavlovic, *Authentication Protocols* teaching notes, Principles of Security, Oxford University, Michaelmas term 2008.