Cryptanalysis of Syverson's Rational Exchange Protocol

Almudena Alcaide, Juan M. Estevez-Tapiador, Julio C. Hernandez-Castro, and Arturo Ribagorda
(Corresponding author: Almudena Alcaide)

Computer Science Department, Carlos III University of Madrid
28911 Leganes, Madrid, Spain (Email: aalcaide@inf.uc3m.es)

(Received Jan. 19, 2006; revised and accepted May 7, 2006 & July 31, 2006)

Abstract

The notion of rational exchange introduced by Syverson in 1998 is a particularly interesting alternative when an efficient scheme for fair exchange is required but the existence of a trusted third party simply cannot be assumed. A rational exchange protocol cannot provide true fairness, but it ensures that rational –i.e. self-interested– parties would have no reason to deviate from the protocol. In this paper, we identify some weaknesses in Syverson's rational exchange protocol which were neither detected by the original author nor by subsequent analysis. After presenting some attacks, we indicate how the scheme should be modified to overcome these vulnerabilities. We also provide a formal analysis of our enhancement using BAN logic.

Keywords: Cryptanalysis, fair exchange, rational exchange, replay attacks

1 Introduction

The problem of how to design a general procedure for two parties to exchange items in a *fair* manner has attracted much recent attention. Interest in this class of protocols stems from its importance in many applications where disputes among parties can occur, such as digital contract signing, certified e-mail, exchange of digital goods and payment, etc. In particular, assurance of fairness is fundamental when the exchanged items include any kind of evidences of non-repudiation, for this constitutes a key service in most of the previously mentioned applications. As a result, fair non-repudiation has experienced an explosion of proposals in recent years (see [4] for an excellent survey).

Roughly, the property of fairness means that no party should reach the end of the protocol in a disadvantageous position, e.g. having sent their item but without having received anything of value in return [1]. Interested readers can find an introduction to the fair-exchange problem in [7].

Formally, there is no protocol according to which a number of parties can exchange items in a fair manner, exclusively by themselves, and assuming that misbehaving parties can take part in the protocol. Pagnia and Gärtner proved this result and provided a formal analysis of the problem in [6]. The underlying idea can be intuitively sketched avoiding technical details: during the protocol execution, one of the parties has eventually to go first in providing their item to the other party. At that point, one of them is in a unfair condition of which a misbehaving party can take advantage. As a result, the simplest protocol that can provide true fairness relies on the use of a trusted third party (TTP). However, recent computing paradigms, such as ad hoc and peer-to-peer networks pose a challenge to this fundamental limit. In many cases, the operation of these systems is based on a complete absence of fixed infrastructures, and it is unrealistic to assume that services such as those provided by a TTP will be available. In fact, in these and in many other scenarios we would probably be forced to renounce to properties such as strong fairness.

It is precisely in this context where notions such as *rationality* become particularly relevant. This concept, widely known by game theorists, was first applied to security protocols by Syverson in 1998 [9]. Informally, a rational exchange protocol cannot provide fairness, but it ensures that rational (i.e. self-interested) parties would have no reason to deviate from the protocol, as misbehaving does not result in any benefit. Since rational exchange protocols provide fewer guarantees, one would expect that they also demand fewer system requirements, so they can be viewed as a trade-off between complexity and true fairness. In particular, rational exchange protocols do not need a trusted third party.

In this paper, we analyze Syverson's rational-exchange protocol [9]. As we will describe in the following sections, there are significant weaknesses in the protocol, and several attacks can be successfully mounted against the scheme. The rest of the paper is organized as follows. In Section 1.1, we introduce Syverson's scheme, while Section 2 serves to present some concepts used in subsequent

$A \rightarrow B: \quad m_1 = (desc_{item_A}, enc(k, item_A), w(k), \sigma_1)$
$B \rightarrow A: \quad m_2 = (item_B, m_1, \sigma_2)$
$A \rightarrow B: \quad m_3 = (k, m_2, \sigma_3)$

where:

$\sigma_1 = sig(k_A^{-1}, (desc_{item_A}, enc(k, item_A), w(k)))$
$\sigma_2 = sig(k_B^{-1}, (item_B, m_1))$
$\sigma_3 = sig(k_A^{-1}, (k, m_2))$

Figure 1: Syverson's rational exchange protocol

analysis. In Section 3, we describe some flaws present in the protocol and how they can be exploited to mount three different attacks. We also analyze the nature of such attacks and some relevant factors. Section 4 is devoted to explain how the protocol can be fixed in order to eliminate previous vulnerabilities, giving a formal proof of our enhancement. Finally, Section 5 summarizes the main conclusions of this work.

1.1 Syverson's Rational Exchange Protocol

For completeness and readability, we first provide a brief review of Syverson's protocol. The scheme is illustrated in Figure 1. A and B denote the two protocol parties, with private keys k_A^{-1} and k_B^{-1}, respectively. We assume that $item_A$ and $item_B$ are the items they would like to exchange, being $desc_{item_A}$ a description of $item_A$. (There is no equivalent description for $item_B$ because the scheme was introduced to serve as a payment protocol, in such a way that $item_B$ has the role of the payment for buying $item_A$). Moreover, $enc(k, m)$ is a symmetric encryption algorithm that encrypts message m with key k. Likewise, $sig(k_i^{-1}, m)$ provides a digital signature on m by using private key k_i^{-1}. Finally, $w(\cdot)$ is a WSBC (Weakly Secret Bit Commitment) function [9]. For our analysis, it suffices to know that $w(x)$ keeps x secret, but it can be broken in acceptable bounds on time.

In step one, A sends B her item $item_A$ in a weakly encrypted form. Next, B sends A her item $item_B$ in return, along with acknowledgement of the first message. Finally, A sends the appropriate key k and acknowledgement of the second message.

We now proceed to analyze some aspects of the protocol just described. These will help in understanding the type of scenarios where the protocol is suitable to use, as Syverson's protocol is not always appropriate.

Note that, at step three, A might fail to send message m_3 or it might not send it for a long time. Furthermore, as B can only disclose the encrypted $item_A$ when the payment has already taken place, A could send a forged $item_A$ and still receive payment in return. The first deterrent against A delaying sending message m_3 is that A gains nothing by doing so, except a bad reputation that could ruin its business. In the case of A sending B the wrong $item_A$, B holds message m_3 as a proof of such misbehavior. However, an important issue arise from both of the previous statements: both participants must exchange during the protocol execution irrevocable evidences to prove the other participant's misbehavior. For example, an scheme on entity A's reputation can only be implemented when it is not possible for B to accuse A of misbehaving if A was honest, and viceversa. A fourth message could be added in which customer B acknowledges timely receipt for message m_3. Likewise, for B to be able to prove in front of an external judging entity that A sent an invalid $item_A$, B must hold irrevocable proof of such a message.

Given the observations above, the author identifies scenarios where the scheme could be used for: (1) If the vendor A is selling relatively low value items, so it is not worth it for the customer (in terms of computational cost or the inconvenience of delay) to break the encryption to recover the item; (2) the vendor A might be selling something that might be of timely and diminishing value, such as short term investment advice or regularly changing lists of bargain items for sale; or (3) the protocol might begin one step earlier with a signed customer request for $item_A$. The vendor A can then take the chance of trading with unknown customers and refuse to service customers who repeatedly fail to pay.

2 Preliminaries

In this section, we briefly introduce some concepts that will be used throughout our analysis.

2.1 A Brief Overview of BAN Logic

Burrows, Abadi and Needham made a significant effort in 1989 defining a logic for the analysis of security protocols [2]. BAN logic is a *logic of beliefs*. An inference process develops from a set of initial beliefs to a set of final goals for each protocol participant. Inference rules are defined as part of the logic.

Next, we introduce a few concepts and two of the BAN logic inference rules, which will be enough for the proofs presented in this paper.

- ***Notation:***
  - $\sharp(M)$: Formula M is fresh, that is, M has not been sent in a message at any time before the current run of the protocol.
  - $P \models M$: Entity P believes M, entity P may act as if M is true.
  - $P \mid\sim M$: P once said M.
- ***Inference Rules:***

1) Freshness Verification Rule (FVR): This rule expresses that if a message is fresh, then the originator of such a message still believes in it:

$$\frac{P \models \sharp(M), \quad P \models Q \mid\sim M}{P \models Q \models M}.$$

2) Encrypted Freshness Verification Rule (EFVR): If a message or part of a message is known to be fresh, then the encrypted message must also be fresh.

$$\frac{\sharp(M)}{\sharp(\{M\}_K)}.$$

Given an entity P and a message M, the statement "P *said* M" ($P \mid\sim M$) implies entity P having said or sent message M at some point in the past. By contrast, the statement "P *believes* M" ($P \models M$) implies entity P to have said or sent M during the current protocol execution –typically taken from the initial point of the protocol run–, so M is *fresh* and A still believes M. This distinction is crucial for our analysis.

2.2 Freshness of Messages and Replay Attacks

Replay attacks consist of the capture of a message –or a piece of a message– that is used at a later time, and probably with a different semantics. *Freshness* of messages is a common and relevant element in security-related protocols, in particular because of its importance as a mechanism to prevent replay attacks. Within the context of a protocol, freshness of a message will guarantee such a message belongs to that specific protocol instance and that it has never been used before in any other instances.

Linking a message to a particular protocol run is commonly obtained by the use of timestamps in messages and timestamping Certification Authorities. Other methods are also implemented, as the use of nonces (randomly created identifiers generated fresh by a participant for each protocol instance [5]), counter values, numbers provided by synchronized pseudo-random number generators, or fresh encryption. See [3] for a detailed description of each of them. However, message replay can take place in many different forms (see [8] for a full classification and taxonomy) and usually more than one of these mechanisms has to be implemented to prevent the protocol from one or another form of replay attack. Freshness of messages is therefore a difficult and very important matter. In any given protocol, the recipient entity of any message should be able to determine whether the message received is fresh. Particularly, our cryptanalysis of Syverson's protocol is based on the impossibility for entity B to determine freshness of message m_1.

3 Cryptanalysis

All messages involved in the protocol are cryptographically signed (see Figure 1). Since such cryptographic algorithms are assumed to be not directly breakable, the primary focus for attackers or penetrators is on the possibility to reuse messages, even when they are not able to read them or to produce them by themselves. By replaying old messages, dishonest parties can impersonate other entities, mislead other participant actions or obtain confidential information.

Syverson's protocol, as defined and described by its author, presents some vulnerabilities, and some attacks can be successfully carried out.

3.1 Observations

The following observations will help in understanding the overall analysis:

1) Message m_1 could be used as a proof of A's misbehavior. Indeed, due to the nature of $w(k)$, if A randomly generates a ciphertext ϵ to include in m_1, A can be penalized whenever the commitment $w(k)$ is broken and k disclosed. Therefore, this kind of misbehavior can always be proven to a judge. In this regard and from B's point of view, the protocol provides a sort of weak fairness. However, also note that m_1 ensures B that A is the author of such a message, but it does not guarantee that A is also the sender of such a message. It is not until step 3 of the protocol that B holds a valid NRO (Non-Repudiation of Origin) token for $item_A$. Therefore, message m_1 could be used to prove that A once generated a forged message, but m_1 cannot be used to prove that A is actually the sender of such a message in the current instance of the protocol.

2) Message m_2 could serve as a NRR (Non-Repudiation of Receipt) token for message m_1 as well as a NRO of $item_B$. B's signature on message m_2 ensures A that B received $item_A$ (weakly encrypted) and that B has proceeded with the sending of $item_B$. Message m_2 could always be used as a proof of B's misbehavior in the protocol.

3) Message m_3 could serve as a NRR token for message m_2 as well as a NRO of $item_A$. A's signature on message m_3 ensures B that A received $item_B$ and that A has generated and sent m_1 with the correct key. Message m_3 could always be used as a proof of A's misbehavior in the protocol. A might not send the third message, or not do it for a long time, but A gains nothing by doing that apart from a poor reputation that could damage their business. The context in which to execute this protocol should then be a regularly repeated scenario.

The protocol, therefore, when rationally executed could provide with rational exchange of non repudiation evidences. However, the non-repudiation evidences would have to be linked to each particular protocol run to serve the purposes of non-repudiation in future disputes. Since A is asked to generate a fresh key k for each run of the

protocol, k could be the unique label to reference each different protocol run and the corresponding evidences. In particular, notice how, although it is possible to determine whether m_2 and m_3 are fresh, this is not the case for m_1 unless some enhancements be made.

3.2 Attack 1

Consider the following scenario, where $P(Q)$ means that party P acts impersonating the role of party Q:

$$\begin{array}{rcll} A & \rightarrow & B & : m_1 = (desc_{item_A}, enc(k, item_A), \\ & & & \quad w(k), \sigma_1) \\ B(A) & \rightarrow & C & : m_1 = (desc_{item_A}, enc(k, item_A), \\ & & & \quad w(k), \sigma_1) \\ C & \rightarrow & B(A) & : m_2 = (item_C, m_1, \sigma_2). \end{array}$$

This attack is based on B impersonating A, sending the same message m_1 to C and receiving $item_C$ in return. B would have to quit the protocol after receiving the payment as she has no key to send to C. Although C has paid a full price for $item_A$, by the time that k is disclosed to C, $item_A$ would be of very little value to C. The customer C could only present message m_1 to prove A misbehaved. However, A will claim that m_1 was never intended for C and that she was not part of such a communication. Indeed, there is nothing in m_1 linking A and C as participants on the same protocol run. To overcome this attack, new restrictions would have to be placed over the communicating network or amendments should be made to the structure of m_1.

3.3 Attack 2

Let us suppose the following simplistic scenario: A is selling an access code to enable the viewing of a football match on a private television network. Let us suppose that A and B carried out a successful Syverson's protocol execution and that they properly exchanged the encrypted access code $enc(k, item_A)$, $item_B$ and the corresponding key k in messages m_{11}, m_{12}, and m_{13}, respectively. The access code that B has bought from A is obviously of timely diminishing value, but B could still have time to impersonate A and sell the access code to other customers, receiving payment in return:

$$\begin{array}{rcll} B(A) & \rightarrow & C & : m_{21} = m_{11} \\ & & & \quad = (desc_{item_A}, enc(k, item_A) \\ & & & \quad w(k), \sigma_A) \\ C & \rightarrow & B(A) & : m_{22} = (payment_C, m_{21}, \sigma_C) \\ B(A) & \rightarrow & C & : m_{23} = m_{13} = (k, m_{12}, \sigma_A). \end{array}$$

In this scenario, by the time C receives message three and realizes that there is a fraud going on, C has no evidence of such a fraud to present in front of a judge and has got the key k to decrypt the football match access code and watch the match. However, A could claim that C is watching a program without a licence and take action against her. If the number of reselling codes is large, the scale of the fraud would make it impractical to pursue each of the individuals watching the match without licence. Furthermore, trying to trail back the origin of such messages would be practically impossible. Again, the nature of the communicating network would have to change or the content of the first message amended.

3.4 Attack 3

If a vendor sends the customer a message m_1 containing garbage (i.e, a ciphertext which does not correspond with the actual $item_A$), the vendor is indeed providing the customer with evidence of such a form of cheating. Message m_1 could be presented to a judge and the vendor would be charged with the appropriate penalty. Such a penalty could greatly exceed the value of the goods, so the vendor is completely discouraged from performing such a scheme. However, the vendor could not be sued and penalized twice for the same offence and, on these terms, a vendor A could carry on sending the forged message m_1 to many others customers, receiving payments in return. These new angry customers would only have message m_1 to blame vendor A. Vendor A would claim that she never sent m_1 to them and that they must have got it from the first resentful customer. As a matter of fact, there will be nothing in m_1 to prove that A is using the same forged message all over again. A's reputation would therefore stay untouched.

4 Fixing the Protocol

Even though the replay attacks one to three described in the previous section correspond to simple deviations from the protocol description, they represent real threats to parties using the scheme to exchange their items. In e-commerce transactions, neither vendor A nor customer B would want to take the risk of being cheated.

However, previous weaknesses can be avoided if a better cryptographic evidence is constructed. This can be done in many ways. Probably the easiest one is just by including the identity of B in m_1, thus linking the message with its intended receiver. Since A is asked to generate a fresh key k for each protocol run, the tuple (k, B, A)[1] could be the unique label to associate each m_1 with the corresponding protocol execution:

$$A \rightarrow B : m_1 = (\mathbf{B}, desc_{item_A}, enc(k, item_A), w(k), \sigma_1)$$

where:

$$\sigma_1 = sig(k_A^{-1}, (\mathbf{B}, desc_{item_A}, enc(k, item_A), w(k))).$$

Note how this modification suffices to prevent attacks one to three. Now, in attack one, an entity C would have sent a payment to a false entity A. With the new structure of message m_1, C would know that m_1 is newly

[1] We assume that A's identity is implicit in m_1, since the message contains A's signature.

formulated by A (since A is asked to create a fresh key k for each instance of the protocol) and that C is the intended recipient. Therefore, A could not claim that it was not part of the protocol run.

In a similar way, this also prevents attack two, for entity B can establish whether the other participant is able to provide key k in the last message of the protocol. Attack three is also easy to prevent, as entity B can tell if the message is an old message that entity A is trying to replay in a new protocol run.

Next is the formalization of all concepts described in this section.

4.1 Formal Analysis

We will formally establish the freshness of messages m_1, m_2, and m_3 after our proposed protocol enhancement. Therefore, any type of replay attack with messages from outside the current execution (old replayed messages) will automatically be rejected, in particular attacks one to three. *Interleaving attacks* (a type of replay attack occurring when two instances of the same protocol are running simultaneously) are not being considered in our analysis, as Syverson's protocol participants are assumed to run only one instance of the protocol at the time. No other form of attack is considered, as messages one to three are digitally signed by algorithms which are assumed to be cryptographically secure.

Below, only those steps of the formal process which are relevant to our enhancement are explicitly shown. Notice how these steps could not have been performed with the original description of the protocol, so freshness of message m_1 could not be guaranteed.

4.1.1 Freshness of m_1

When entity B receives m_1, B knows A's public key and is able to verify A's signature on m_1. Once B verifies the signature, B can be sure that the originator of that message was entity A. In BAN logic notation, we would express:

$$B \models A \mid\sim m_1. \tag{1}$$

Furthermore, B can see their name as part of the message so B is convinced she is the intended recipient. The item $desc_{item_A}$ serves B to identify m_1 as unique. Entity A has signed a message where $item_A$ has been encrypted and B is the intended recipient. Entity B believes this message could not have been used in any other instances of the protocol of which she was not part. Therefore, the combined tuple $(B, desc_{item_A})$, which is part of message m_1, is fresh: $B \models \sharp(B, desc_{item_A})$. Then, the following formula can be inferred applying the EFVR:

$$B \models \sharp(m_1). \tag{2}$$

Note that if B was buying the same $item_A$ twice, then entity B would have to verify that the two message components: B and $enc(k, item_A)$ were never bound together in any of the previous instances. Remember that, entity A is forced to generate a new key k for each new run.

Therefore, in any given case, applying FVR to Equations (1) and (2) we obtain that:

$$B \models A \models m_1,$$

which ensures freshness of m_1.

4.1.2 Freshness of m_2

When entity A receives m_2, A knows B's public key and is able to verify B's signature on m_2. Once A verifies the signature, A can be sure that the originator of that message was entity B. In BAN logic notation we would express:

$$A \models B \mid\sim m_2. \tag{3}$$

Moreover, A can see message m_1 as part of message m_2. Entity A generated m_1 as step one of the protocol so A believes m_2 is fresh as it could not have been generated in any other previous instances of the protocol. In BAN logic notation we have:

$$A \models \sharp(m_2). \tag{4}$$

Now, applying FVR to Equations (3) and (4), we obtain a proof of freshness for m_2:

$$A \models B \models m_2.$$

4.1.3 Freshness of m_3

This part of the formal verification is exactly the same as for the freshness of message m_2. Therefore, mirroring the previous steps, we can conclude that m_3 is also fresh:

$$B \models A \models m_3.$$

5 Conclusions

In this paper, we have demonstrated how Syverson's scheme suffers from some weaknesses due to an inappropriate design of the cryptographic evidences. Our attacks show up that the protocol can lead to undesired situations for any of the two parties involved in the exchange. The aforementioned vulnerabilities have not been pointed out before.

We have also suggested how to fix the scheme and we have given a formal security proof of the enhancement. The proof is based on guaranteeing freshness of all protocol messages, thus ensuring rejection of all forms of replay attacks.

References

[1] N. Asokan, *Fairness in electronic commerce*, Ph.D. Thesis, University of Waterloo, 1998.

[2] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication," *ACM Transations on Computer Systems*, vol. 8, no. 1, pp. 18-36, 1990.
[3] L. Gong, "Variations on the Themes of Message Freshness and Replay," in *Proceedings of the IEEE Computer Security Foundations Workshop VI*, pp. 131-136, June 1993.
[4] S. Kremer, O. Markowitch, and J. Zhou, "An intensive survey of fair non-repudiation protocols," *Computer Communications*, vol. 25, no. 17, pp. 1606-1621, 2002.
[5] R. M. Needham, and M. D. Schroeder, "Using encryption for authentication in large networks of computers," *Communications of the ACM*, vol. 21, no. 21, pp. 993-999, Dec. 1978.
[6] H. Pagnia, and F.C. Gärtner, *On the Impossibility of Fair Exchange without a Trusted Third Party*, Darmstadt University of Technology, Department of Computer Science, Technical Report TUD-BS-1999-02. Mar. 1999.
[7] H. Pagnia, H. Vogt, and F.C. Gärtner, "Fair exchange," *The Computer Journal*, vol. 46, no. 1, Jan. 2003.
[8] P. Syverson. "A Taxonomy of Replay Attacks," in *Proceedings of the 7th IEEE Computer Security Foundations Workshop*, pp. 187-191, 1994.
[9] P. Syverson, "Weakly secret bit commitment: Applications to lotteries and fair exchange," in *Proceedings of the 11th IEEE Computer Security Foundations Workshop*, pp. 2-13, 1998.

Almudena Alcaide Raya is Assistant Professor at the Computer Security Group of the Computer Science Department of Carlos III University of Madrid. She has a B.Sc. in Mathematics by Complutense University of Madrid and a M.Sc. in Advanced Computing by Kings College of London. Currently, she is a Ph.D. student at the Computer Science Department of Carlos III University of Madrid. Her work is focused on formal methods applied to the design and analysis of cryptographic protocols. In particular, her most recent research activity is related to applying Game Theory results to security protocols, and how these methods can assist in developing automated protocol verification tools and techniques.

Arturo Ribagorda is Full Professor at Carlos III University of Madrid, where he is also the Head of the Computer Security Group and currently acts as the Director of the Computer Science Department. He has a M.Sc. in Telecommunications Engineering and a Ph.D. in Computer Science. He is one of the pioneers of computer security in Spain, having more than 25 years of research and development experience in this field. He has authored 4 books and more than 100 articles in several areas of information security.

Juan M. Estevez-Tapiador is Associate Professor at the Computer Science Department of Carlos III University of Madrid. He holds a M.Sc. in Computer Science from the University of Granada (2000), where he obtained the Best Student Academic Award, and a Ph.D. in Computer Science (2004) from the same university. His research is focused on cryptography and information security, especially in formal methods applied to computer security, design and analysis of cryptographic protocols, steganography, and some theoretical aspects of network security. In these fields, he has published around 40 papers in specialized journals and conference proceedings. He is member of the program committee of several conferences related to information security and serves as regular referee for various journals.

Julio C. Hernandez-Castro is Associate Professor at the Computer Science Department of Carlos III University of Madrid. He has a B.Sc. in Mathematics, a M.Sc. in Coding Theory and Network Security, and a Ph.D. in Computer Science. His interests are mainly focused in cryptology, network security, steganography and evolutionary computation.