Establishing and Preventing a New Replay Attack on a Non-Repudiation Protocol

Carla Muntean*, Reiner Dojen*, Tom Coffey*
*Department of Electronic & Computer Engineering,
University of Limerick, Ireland.
E-Mail: carla.muntean@ul.ie, reiner.dojen@ul.ie, tom.coffey@ul.ie

Abstract

Non-repudiation is a security service concerned with preventing a denial by one of the principals involved in a communication about having participated in this communication. In this paper, the Zhou Gollmann non-repudiation protocol is analyzed using an automated logic-based verification engine. As a result of this analysis a weakness in the protocol is discovered. Based on this weakness, a new replay attack on the Zhou Gollmann protocol is presented. In this attack, an intruder can incorrectly convince a principal to have successfully performed a new message exchange. As a consequence, the intruder can impersonate legitimate principals. The weakness leading to the attack is analyzed in detail and amendments to the protocol are proposed that prevent the presented attack. Further, formal verification of the amended protocol provides strong confidence in its correctness and effectiveness.

Keywords:
Non-repudiation, fairness, replay attack, Zhou-Gollmann, freshness

I.INTRODUCTION

Through the explosion of Communication Technology over the last decade, electronic transactions are increasingly concerned with confidential data. Security protocols are an important component in providing trusted services, such as key distribution, authentication and non-repudiation. Non-repudiation is concerned with preventing an entity to deny its participation in a message exchange. Thus, non-repudiation protocols provide for undeniable data exchange between two or more principals [1], [2], [3].

Formal methods provide means to verify such protocols and add confidence in their correctness. The use of modal logic has been shown to be an effective formal method in detecting flaws in the design of security protocols [4], [5], [6].

This paper analyses the Zhou Gollmann non-repudiation protocol. The protocol is reviewed and its formal verification is detailed. The results of this analysis reveal a weakness in the protocol and a new attack exploiting this weakness is presented. Amendments to the protocol are proposed to protect it against the introduced attack. Formal verification of the amended protocol provides strong confidence in its correctness and effectiveness.

II. FAIR NON-REPUDIATION PROTOCOLS

Non-repudiation is a security service concerned with preventing a denial or repudiation by one of the principals involved in a communication about having participated in this communication. A non-repudiation protocol has to ensure that proper non-repudiation evidences are delivered to the involved principals. In case of a dispute, a judge can evaluate this evidence and take an objective decision in favor of one of the parties. The following types of non-repudiation are distinguished:

- **Non-repudiation of origin (NRO)** provides the recipient with the evidence of the fact that the message has been sent during the protocol run; it is generated by the originator and held by the recipient
- **Non-repudiation of receipt (NRR)** provides the originator with the evidence of the fact that the message has been delivered properly during the protocol run; it is generated by the recipient and held by the originator of the message
- **Non-repudiation of submission (NRS)** provides evidence that the originator submitted the message for delivery. The service applies only when a TTP is involved. It is generated by the delivery agent and it will be sent to the originator of the message
- **Non-repudiation of delivery (NRD)** provides evidence that the recipient received the message. This service applies only when a TTP is involved. It is generated by the delivery agent and it will be sent to the originator of the message.

Additionally, a non-repudiation protocol must ensure *timeliness* in order to be considered practical. Timeliness guarantees that the

participating entities always finish the protocol after a certain finite amount of time. It also avoids situations where an entity does not know whether a protocol run is finished or not and consequently needs to keep the session open to assure fairness.

Another important aspect in non-repudiation is *fairness*, i.e. not to place any entity in an advantageous position over the other. A non-repudiation protocol is called *fair* if, at the end of the protocol execution, either the originator has the evidence of receipt for the message *m* and the recipient has the evidence of origin of the same message, or neither has any valid evidences.

Many non-repudiation protocols make use of one or multiple trusted third parties (TTP) [1], [3], [7], [8], [9] which can be categorized as follows:

- An **inline TTP** is acting as a delivery authority and is involved in all message exchanges.
- An **online TTP** intervenes in each session, but not in each message transmission.
- An **offline TTP** intervenes only in case of a dispute – either incorrect behaviour of a dishonest entity or a network error.

III. THE ZHOU GOLLMANN NON-REPUDIATION PROTOCOL

The Zhou Gollmann protocol (ZG) is a non-repudiation protocol which uses an online trusted third party. The notations used in describing the protocol are presented in Table 1 and Fig. 1 outlines the steps of the ZG protocol.

Table 1: Notations used for the ZG protocol

A	The originator of the message
B	The recipient of the message
TTP	Trusted Third Party
data	Message that A wants to transmit to B
KxPub	Public key of entity *x*
KxPriv	Private key of entity *x*
Kab	Session key generated by A
Nx	Nonce belonging to entity *x*
dataNRO	Flag indicating message contains NRO
dataNRR	Flag indicating message contains NRR
dataSUB	Flag indicating message contains NRS
dataCON	Flag indicating message contains confirmation of transaction
{data}K	Encryption of data *data* using key *K*
A->B:m	Principal A sends message *m* to B

Principal A initiates the protocol by sending to B the flag *dataNRO*, B's identity, the nonce *Na* and the message *dataA* encrypted with session key *Kab* as well as all these signed with A's private key. *Na* is a unique label identifying the protocol session.

B can decrypt the message sent by A using A's public key, but cannot decrypt the message *dataA* at this time, as B does not yet possess the session key *Kab*. B replies by sending to A the flag *dataNRR*, A's identity, the nonce *Na* and the encrypted message *{data}Kab* as well as all these signed with B's private key.

```
1. A -> B: dataNRO,B,Na,{dataA}Kab,
     {dataNRO,B,Na,{dataA}Kab}KaPriv
2. B -> A: dataNRR,A,Na,{dataA}Kab,
     {dataNRR,A,Na,{dataA}Kab}KbPriv
3. A -> TTP: dataSUB,B,Na,Kab,
     {dataSUB,B,Na,Kab}KaPriv
4. TTP -> B: dataCON,A,B,Na,Kab,
     {dataCON,A,B,Na,Kab}KttpPriv
5. TTP -> A: dataCON,A,B,Na,Kab
     {dataCON,A,B,Na,Kab}KttpPriv
```

Figure 1: Steps of the ZG protocol

On reception of this message, A confirms that the correct nonce *Na* is used and then sends to the TTP a message containing the flag *dataSUB*, B's identity, the nonce *Na* and the session key *Kab* as well as all these signed by its private key.

The TTP decrypts A's message using A's public key and now possesses the session key *Kab*. Then the TTP compares the nonce/key pair sent by A against a list of previously used pairs. If a unique pair has been used, the TTP accepts the message as a genuine request by A. Otherwise the message is considered a replay and will be discarded.

In the next two steps the TTP sends a confirmation message to both principals. This confirmation message is intended to complete non-repudiation of origin and receipt. These messages contain the confirmation flag *dataCON*, the identities of A and B, the nonce Na and the session key *Kab* as well as all these signed by the TTP.

In case of a dispute, if principal A claims to have sent the message to B, the judge will ask A to provide the message that has been sent and the non-repudiation of receipt for this message. The evidence that A has to give is composed of the non-repudiation of receipt provided by B and the confirmation message provided by the TTP. If all the evidence is proven to be valid, the judge closes the case and declares A's assertion as correct. On the other hand, if A falsely claims to have sent the message, A will not be in possession of complete and valid evidence and A's claim will be refuted.

Similar for principal B: If B claims to have received a message from A the judge will request B to provide the message and the non-repudiation of origin. The evidence that B has to present is composed of the non-repudiation of origin for the message being sent (provided by A) and the confirmation message provided by the TTP. If B has all this evidence, the judge will assert B's claim to be correct. Again, if B falsely claims to have received a message from A, B will no be able to provide the complete and valid evidence.

IV. FORMAL VERIFICATION

The technique of logic-based formal verification is accredited largely to Burrow, Abadi and Needham, developers of the well known BAN logic [4]. Several logics, such as [5], [10], [11] have been developed based on the BAN logic. In general, logic-based formal verification involves the following steps:

1) Formalization of the protocol: requires specifying the protocol in the language of the logic by expressing each protocol step as a logical formula.
2) Specification of the initial assumptions: description of the initial beliefs and possessions held at the beginning of the protocol run by the principals involved in the communication process.
3) Specification of protocol goals: goals of the protocol are expressed in the language of the logic in terms of possessions and beliefs held by the principals at the end of the protocol run.

The objective of the logical analysis is to verify whether the expressed goals can be proven and justified from the initial assumptions and steps. If all the goals are proven to be correct, the protocol can be considered correct. If at least one of the goals fails the protocol will be verified as 'false'. This failure can point to a weakness in the protocol design or an error in the formalized protocol specification. In case of a weakness, the protocol should be redesigned and verified again.

A. CDVT – A Logic-Based Verification Engine

The CDVT verification engine is an automated system that implements a modal logic of knowledge and belief using Layered Proving Trees [12] and a parser to read in the protocol specification from a text file, which contains the formal specification of the protocol to be verified as follows [13]: elements follow regular expressions as expressed in *Table 2,* data components are build according to *Table 3* and statements construction rules are defined in *Table 4*.

Table 2: Atomic Units of Textual Grammar

Component	Regular Expression
Principal	[AB-EIJLMOQRSU-Z] [A-Za-z_0-9_]*
Trusted Principal	TTP[A-Za-z0-9_]*
Sym. Key	K[a-z][a-zA-Z0-9_]*
Public Key	K[a-z][A-Za-z0-9_]*Pub
Private Key	K[a-z][A-Za-z0-9_]*Priv
Nonce	N[a-z][A-Za-z0-9_]*
Timestamp	TS[a-z][A-Za-z0-9_]*
Function	F[A-Za-z0-9_]*
Hash	H[A-Za-z0-9_]*
Binary Data	[a-z][A-Za-z0-9_]*

Table 3: Composite Data Construction

Composite Data	Textual Representation
Concatenation	*Data,Data*
Group Element	(*Data*)
Symmetric Encryption	{*Data*}*Data*
Public Key Encryption	{*Data*}Kpub
Private Key Encryption	{*Data*}Kpriv
Function of Data	F(*Data*)
Hash of Data	H(*Data*)

Table 4: Statement Construction

Principal *operator* at[i] *Data*
Principal know at[i] *Statement*
Principal believe at[i] *Statement*
Principal know at[i] NOT (*Statement*)
Principal believe at[i] NOT (*Statement*)
(*Statement*)
(*Statement* AND *Statement*)
(*Statement* IMPLY *Statement*)

V. VERIFICATION OF THE ZHOU GOLLMANN PROTOCOL

This section uses the CDVT verification engine to establish the correctness of the ZG protocol. In order to verify a protocol its initial assumptions, steps and goals must be translated into the language of the employed verification logic.

A. Initial assumptions

The specification of the initial assumptions of the ZG protocol is presented in Fig. 2.

```
A1:  A possess at[0] KttpPub;
A2:  A possess at[0] KbPub;
A3:  A know at[0]
     B possess at[0] KbPriv;
A4:  A know at[0]
     TTP possess at[0] KttpPriv;
A5:  A possess at[0] Na;
A6:  A know at[0] (not(Zero possess
      at[0] Na));
A7:  B possess at[0] KttpPub;
A8:  B possess at[0] KaPub;
A9:  B know at[0] TTP possess at[0]
      KttpPriv;
A10: B know at[0] A possess
      at[0] KaPriv;
A11: TTP possess at[0] KaPub;
A12: TTP possess at[0] KbPub;
A13: TTP know at[0] A possess at[0]
     KaPriv;
A14: TTP know at[0] B possess at[0]
      KbPriv;
A15: TTP know at[0] (not(Zero possess
      at[0] (dataSUB,B,Na,Kab)));
```

Figure 2: ZG protocol - initial assumptions

Assumptions A1 and A2 express the fact that before the start of the protocol, A possesses both

TTP's and B's public keys. A3 and A4 state that A is aware both B and TTP possess their own private keys before the protocol starts. This binds the public keys to the identity of the corresponding principal. A5 specifies that A possesses the nonce Na and assumption A6 states that A knows that no other principal can possess this nonce at the beginning of the protocol run.

A7 and A8 express the fact that B is in possession of both A's and TTP's public key and assumptions A9 and A10 specify that B knows before the protocol starts that both A and TTP are in possession of their own private keys.

Assumptions A11 and A12 state that TTP is in possession of both A and B's public keys. A13 and A14 declare that TTP knows both A and B are in possession of their private keys before the protocol run. A15 expresses the fact that the TTP will detect and discard messages that contain previously used nonce/key pairs.

B. Protocol Steps

The formalized steps for the ZG are detailed in Fig. 3. These are a direct translation of the protocol steps as discussed in Section III.

```
S1: B receive at[1] dataNRO,B,Na,
     {dataA}Kab,{dataNRO,B,Na,
     {dataA}Kab}KaPriv;
S2: A receive at[2] dataNRR,A,Na,
     {dataA}Kab,{dataNRR,A,Na,
     {dataA}Kab}KbPriv;
S3: TTP receive at[3] dataSUB,B,Na,
     Kab,{dataSUB,B,Na,Kab}KaPriv;
S4: B receive at[4] dataCON,A,B,Na,
     Kab,{dataCON,A,B,Na,Kab}KttpPriv;
S5: A receive at[5] dataCON,A,B,Na,
     Kab,{dataCON,A,B,Na,Kab}KttpPriv;
```

Figure 3: ZG protocol - steps

C. Protocol Goals

The goals of the ZG protocol are presented in Fig. 4. Goals G1 states that B in possession of the evidence of origin (Note that this evidence is incomplete without the confirmation message).

Similar, goals G2, G3 determine that A is in possession of the evidence of receipt from B (incomplete without confirmation message) and that B is indeed the source of the message.

Goals G4 to G6 are related with the fact that TTP has obtained confirmation of submission of key *Kab* by A and that the nonce/key pair hasn't been used in a previous protocol run. G7 to G9 establish that B receives the message that TTP sent in step 3 and that B possesses the session key *Kab* and the clear text message dataA. Goals G10 and G11 express that B knows that the message received at step 4 of the protocol run was sent by TTP and that this message is fresh.

G12 to G14 state that A has now the complete evidence of receipt for the message sent to B. G12 expresses A's possession of the confirmation of the session key sent by TTP. G13 determines that A knows that the TTP is indeed the source of message S5 and G14 describes that A is able to establish freshness of S5.

```
G1:  B possess at[1]{dataNRO,B,Na,
       {dataA}Kab}KaPriv;
G2:  A possess at[2] {dataNRR,A,Na,
       {dataA}Kab}KbPriv;
G3:  A know at[2] (B send at [2]
       {dataNRR,A,Na,
       {dataA}Kab}KbPriv);
G4: TTP possess at[3]
       {dataSUB,B,Na,Kab}KaPriv;
G5: TTP know at[3] (A send at[3]
       {dataSUB,B,Na,Kab}KaPriv);
G6: TTP know at[3](not(Zero send at[0]
       {dataSUB,B,Na,Kab}KaPriv));
G7:  B possess at[4]
       {dataCON,A,B,Na,Kab}KttpPriv;
G8:  B possess at[4] Kab;
G9:  B possess at[4] dataA;
G10: B know at[4] (TTP send at[4]
       {dataCON,A,B,Na,Kab}KttpPriv);
G11: B know at[4] (not(Zero send at[0]
       {dataCON,A,B,Na,Kab}KttpPriv));
G12: A possess at[5]
       {dataCON,A,B,Na,Kab}KttpPriv;
G13: A know at[5] (TTP send at[5]
       {dataCON,A,B,Na,Kab}KttpPriv);
G14: A know at[5] (not(Zero send at[0]
       {dataCON,A,B,Na,Kab}KttpPriv));
```

Figure 4: ZG protocol - goals

D. Results of the verification

The verification result of the ZG protocol is presented in Fig. 5. Goals G1 to G9 and G12 to G14 are established, but G10 and G11 are failing, as principal B is unable to ensure validity of the confirmation evidence received in the protocol step S4. This confirmation evidence is required to complete the evidence of origin received in step S2.

Further investigation of the attempted verification of goals G10 and G11 reveals that failure is caused by B's inability to establish freshness of the message received from TTP in protocol step S4. This step includes the confirmation evidences created by the TTP, which is required by B to complete the non-repudiation on origin evidence on the message send by A.

As message S4 doesn't contain anything that B can recognize as being fresh, B is unable to bind the received message to the current protocol run. Therefore, B cannot distinguish a new valid message from an old and potentially compromised message replayed by an intruder. Consequently, the protocol is susceptible to a replay attack in which an intruder can trick principal B into accepting messages from previous protocol runs.

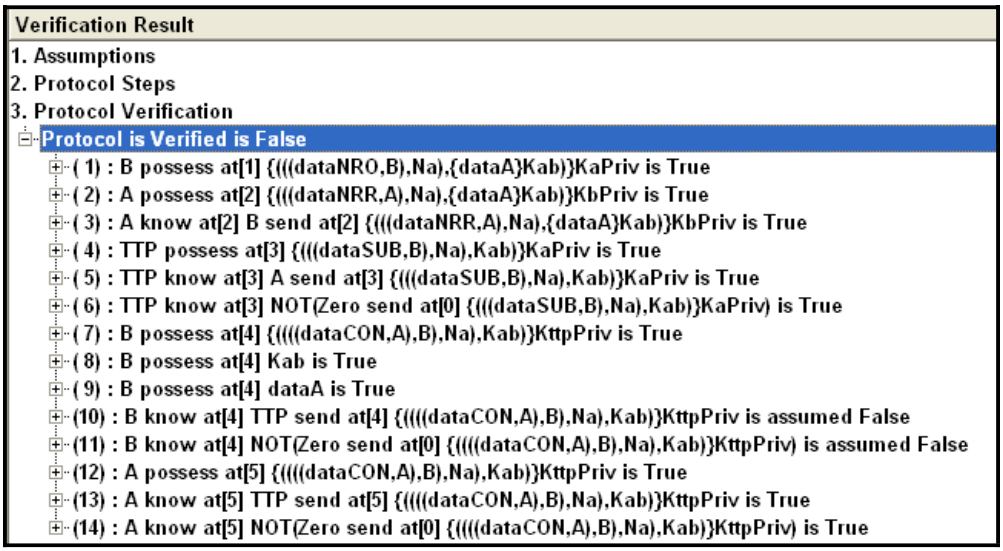

Figure 5: Verification results of ZG protocol

VI. A NEW ATTACK ON THE ZHOU GOLLMANN PROTOCOL

In this section a new attack on the ZG protocol is presented. This attack exploits the weakness of the protocol as revealed in the previous section.

The structure of this new attack is detailed in Fig. 6, where "I(A)" and "I(TTP)" notate an intruder taking on the role of principal A and the TTP, respectively. This attack assumes that the protocol has been previously executed successfully by principals A and B and that the intruder has recorded the message exchanged in this previous protocol run. During the attack, the intruder takes on the roles of both, principal A and the TTP.

```
1.  I(A) -> B: dataNRO,B,Na,
      {dataA}Kab,{dataNRO,B,Na,
      {dataA}Kab}KaPriv
2.  B -> I(A): dataNRR,A,Na,
      {dataA}Kab,{dataNRR,A,Na,
      {dataA}Kab}KbPriv
3.  omitted
4.  I(TTP) -> B: dataCON,A,B,Na,Kab,
      {dataCON,A,B,Na,Kab}KttpPriv
5.  omitted
```

Figure 6: A new attack on the ZG protocol

The attack begins with the intruder impersonating principal A by sending to B the initial message from the previous recorded exchange. B, who assumes that the message is sent by principal A, replies with the second step as per the protocol specification. The intruder again takes on the role of A and intercepts the message. The third step is omitted as the intruder impersonates both, principal A and the TTP. In the fourth step the intruder masquerades as the TTP and sends to B the fourth message from the recorded session. Analogous to step 3, step 5 is also omitted.

At the end of such a protocol run, B is convinced to have performed a new message exchange with principal A. However, neither principal A nor the TTP have been involved in this new message exchange, and, thus, have no knowledge of the repeated protocol run.

VII. PREVENTING REPLAY ATTACKS

Data freshness is an essential security requirement that facilitates the ability of participating principals to bind the exchanged messages to the current protocol run. In general, data (particularly encrypted expressions) generated for one protocol session should not be valid in any other session of the protocol. This ensures that an attacker who gains access to messages (or parts of messages) cannot use these in a replay attack. This can be achieved by including suitable freshness identifiers in the structure of the messages. Such freshness identifiers can be based on timestamps or nonces.

A. Timestamps

A timestamp denotes the date and time at which a certain event occurred. When timestamps are used in security protocols to provide data freshness, this event usually is the time at which a principal sends or creates a message. For example, assume that at time t_1 principal A creates a message *m* that contains a timestamp TS_A (representing time t_1) to principal B. On receipt of the message *m* B will compare the timestamp TS_A against the current time t_2. If the difference between t_2 and TS_A is below an agreed threshold ε, i.e. $t2\text{-}TS_A < \varepsilon$, then the message *m* is considered fresh. On the other hand, if this difference is greater than ε, then the message is considered to be expired and will be discarded.

Security protocols based on timestamps have the advantage of being stateless, as each request is unrelated to any previous request. Also, they typically require fewer messages than protocols based on nonces. However, timestamps do also have a number of disadvantages [14], [15], [16]: Most notably, the use of timestamps requires that all principals have reliable and synchronised clocks. Further, any timestamp based protocol potentially allows the replay of messages until the end of the valid time interval, resulting in a window attack. Extra measurements, such as storing messages until they are expired, are required to prevent such window attacks.

B. Nonces

Nonces are once-off random numbers used to ensure data freshness. They can also be considered to be unique labels that identify a certain session of a protocol. The sequence of nonces generated by a principal should be unpredictable and a principal should use every nonce only once. To ensure freshness of a message, a principal must first distribute its own, newly generated nonce to other principals. For example, if principal A sends its fresh nonce Na to principal B, B can subsequently reply with a message containing Na. A can now verify that this message belongs to the current protocol run by ensuring that the correct value of Na is used.

Security protocols based on nonces have the advantage that they do not require clock synchronization and that no window attack is possible. However, the difficulty of generating proper random numbers is one of the major disadvantages of nonces [17], [18]. Another drawback is that nonce-based security protocols require more messages as timestamp-based ones, as the nonce must be distributed before it can be used to ensure freshness.

VIII. AMENDING THE ZHOU GOLLMANN PROTOCOL

As demonstrated above the ZG protocol has a freshness weakness that can be exploited by a replay attack. Cause of the weakness is B's inability to establish freshness of the message received at step S4.

In order to fix the flaw, B needs to have a proof in the message received from the TTP to confirm that this message belongs to the current protocol run. To avoid the problem of reliable, synchronized clocks, it is proposed to include a second nonce *Nb*, which is generated by principal B, in the exchanged messages: In the second step B submits this nonce as part of its message to A, who will forward it as part of step 3 to the TTP. Inclusion of both nonces *Na* and *Nb* into the confirmation evidence distributed by the TTP ensures that both, principals A and B, are able to establish freshness of this evidence. The protocol steps of the proposed amended protocol are presented in Fig. 7. Operation of the amended protocol is analogous to the original protocol as detailed in Section III, but includes the nonce *Nb* in steps 2 to 5.

```
1. A -> B: dataNRO,B,Na,{dataA}Kab,
     {dataNRO,B,Na,{dataA}Kab}KaPriv
2.  B -> A: dataNRR,A,Na,Nb,
     {dataA}Kab,{dataNRR,A,Na,
     {dataA}Kab}KbPriv
3. A -> TTP: dataSUB,B,Na,Nb,Kab,
     {dataSUB,B,Na,Nb,Kab}KaPriv
4. TTP -> B: dataCON,A,B,Na,Nb,Kab,
     {dataCON,A,B,Na,Nb,Kab}KttpPriv
5. TTP -> A: dataCON,A,B,Na,Nb,Kab
      {dataCON,A,B,Na,Nb,Kab}KttpPriv
```

Figure 7: Steps of the amended ZG protocol

IX. VERIFICATION OF THE AMENDED ZHOU GOLLMAN PROTOCOL

In this section, the amended ZG protocol is formally verified to establish its correctness and effectiveness.

A. Initial assumptions

The initial assumptions of the amended ZG protocol are essentially identical with the ones of the original ZG protocol. However, the assumptions shown in Fig. 8 are additionally required to capture knowledge about the nonce *Nb*. The modified assumption A15 reflects the inclusion of *Nb* in the confirmation evidence generated by the TTP. A16 specifies that B possesses the nonce *Nb* at the start of the protocol and assumption A17 states that B knows that no other principal can possess this nonce at the beginning of the protocol run.

```
A15: TTP know at[0] (not(Zero possess
       at[0] (dataSUB,B,Na,Nb,Kab)));
A16: B possess at[0] Nb;
A17: B know at[0] (not(Zero possess
       at[0] Nb));
```

Figure 8: Amended ZG - additional initial assumptions

B. Protocol steps

Formalisation of the protocol steps is analogous to the original ZG protocol, but includes addition of the nonce *Nb* in messages S2 to S5. Fig. 9 illustrates the formalised steps of the proposed fixed protocol.

```
S1: B receive at[1] dataNRO,B,Na,
      {dataA}Kab,{dataNRO,B,Na,
      {dataA}Kab}KaPriv;
S2: A receive at[2] dataNRR,A,Na,Nb,
      {dataA}Kab,{dataNRR,A,Na,Nb
      {dataA}Kab}KbPriv;
S3: TTP receive at[3] dataSUB,B,Na,Nb,
      Kab,{dataSUB,B,Na,Nb,Kab}KaPriv;
S4: B receive at[4] dataCON,A,B,Na,Nb,
      Kab,{dataCON,A,B,Na,Nb,
      Kab}KttpPriv;
S5: A receive at[5] dataCON,A,B,Na,Nb,
      Kab,{dataCON,A,B,Na,Nb,
      Kab}KttpPriv;
```

Figure 9: Amended ZG - steps

C. Protocol Goals

The goals of the fixed protocol are essentially identical to the goals of the original ZG protocol. However, they need to reflect the changes made to the exchanged messages (cf. Fig. 10).

```
G1:  B possess at[1]{dataNRO,B,Na,
      {dataA}Kab}KaPriv;
G2:  A possess at[2] {dataNRR,A,Na,Nb,
      {dataA}Kab}KbPriv;
G3:  A know at[2] (B send at [2]
      {dataNRR,A,Na,N,
      {dataA}Kab}KbPriv);
G4: TTP possess at[3]
      {dataSUB,B,Na,Nb,Kab}KaPriv;
G5: TTP know at[3] (A send at[3]
      {dataSUB,B,Na,Nb,Kab}KaPriv);
G6: TTP know at[3](not(Zero send at[0]
      {dataSUB,B,Na,Nb,Kab}KaPriv));
G7:  B possess at[4]
      {dataCON,A,B,Na,Nb,Kab}KttpPriv;
G8:  B possess at[4] Kab;
G9:  B possess at[4] dataA;
G10: B know at[4] (TTP send at[4]
      {dataCON,A,B,Na,Nb,
      Kab}KttpPriv);
G11: B know at[4] (not(Zero send at[0]
      {dataCON,A,B,Na,Nb,
      Kab}KttpPriv));
G12: A possess at[5]
      {dataCON,A,B,Na,Nb,Kab}KttpPriv;
G13: A know at[5](TTP send at[5]
      {dataCON,A,B,Na,Nb,
      Kab}KttpPriv);
G14: A know at[5](not(Zero send at[0]
      {dataCON,A,B,Na,Nb,
      Kab}KttpPriv);
```

Figure 10: Amended ZG - goals

D. Results of the verification

Verification of the fixed protocol using the CDVT engine produces the results shown in Fig. 11. As all protocol goals are proven to be true, the fixed protocol is verified successfully.

In the proposed fix, the nonce *Nb* has been introduced. This nonce is generated freshly by and issued by principal B for each protocol run. As now the confirmation evidence distributed by the TTP contains the two nonces *Na* and *Nb*, both principals are able to determine freshness of the confirmation evidence. Freshness of this evidence ensures that the received messages belongs indeed to the current protocol run and is not a replay from a previous session. Consequently, an intruder cannot successfully replay recorded messages:

Any attempt to replay message S1 will fail, as the intruder will be unable to create the correct message component "{dataSUB, B, Na, Nb, Kab}KaPriv" required for message 3. Further, as each honest principal will choose a fresh and unique nonce for each protocol run, it is impossible for an intruder to obtain this component from any other protocol run. An attempt to replay message 3 as well to avoid construction of this component will also fail: B will detect the usage of an incorrect *Nb* in the confirmation evidence. Analogous, any attempt of impersonating principal B by replaying message S2 will fail as an intruder cannot create the required message with the current *Na*. Similarly, an intruder cannot impersonate the TTP. As the confirmation evidence contains a fresh component for each principal (*Na* and *Nb*, respectively), neither message S4 nor S5 can be replayed. Thus, each principal is able to detect an attempted replay.

X. CONCLUSIONS

This paper concerned the correctness of non-repudiation protocols. The concept of non-repudiation was discussed and the non-repudiation protocol by Zhou and Gollmann was reviewed.

A formal verification of the ZG protocol using an automated logic-based verification engine was presented and the verification results were discussed. The analysis of these results revealed that the ZG protocol suffers from a freshness weakness. A new replay attack on the ZG protocol that exploits the revealed weakness has been introduced. This attack allows an intruder to trick principal B into accepting messages from previous protocol runs. Consequently, principal B is incorrectly convinced to have performed a new message exchange with principal A.

A new protocol has been proposed that fixes the freshness weakness. The details of the fixed protocol and its formal verification have been presented. The results of the verification are analysed and the effectiveness of the new protocol is discussed, where it is demonstrated that the protocol is immune against replay attacks. Overall, the formal analysis of the fixed protocol provide strong confidence in its correctness.

Acknowledgements:
This work was partially funded by Science Foundation Ireland - Research Frontiers Programme SFI 07/RFP/CMSF 631.

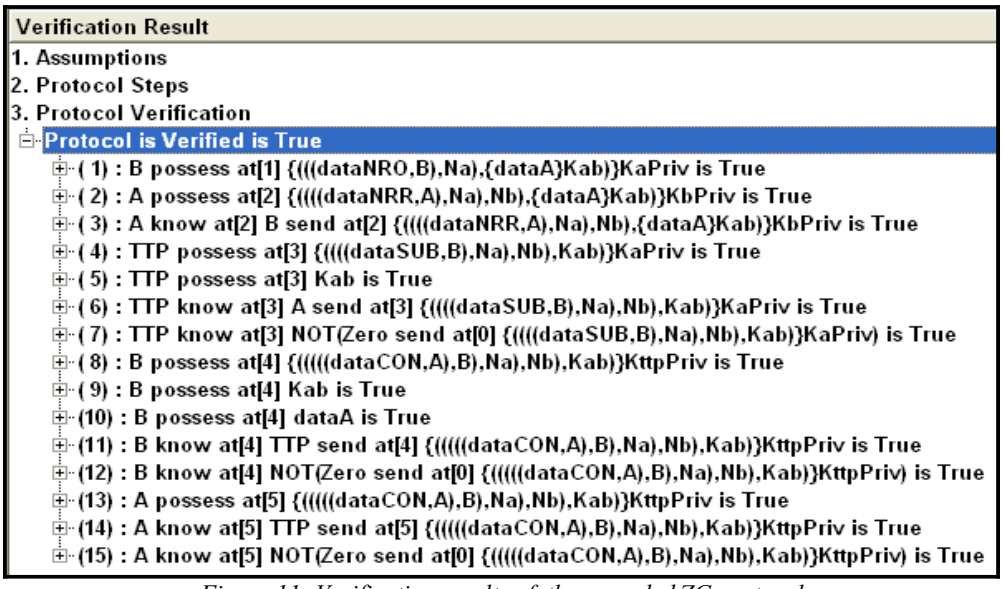

Figure 11: Verification results of the amended ZG protocol

REFERENCES

[1] J. Zhou and D. Gollmann, "A fair non-repudiation protocol", Proceedings of 1996 IEEE Symposium on Security and Privacy, IEEE Computer Security Press, Oakland, California, May 1996, pp 55-61

[2] O. Markowitch and S. Kremer, "An Optimistic Non-repudiation Protocol with Transparent Trusted Third Party", Proceedings of ICIS 2001, Lecture Notes in Computer Science, Vol. 2200, Springer-Verlag, 2001, pp. 363-378

[3] T. Coffey and P. Saidha, "Non-repudiation with Mandatory Proof of Receipt", Computer Communication Review, Vol.26, No.1, January 1996, pp. 6-17

[4] M. Burrows, M. Abadi, R. Needham, "A logic of authentication", ACM Transactions on Computer Systems TOCS, Vol. 8, No. 1, 1990, pp. 18-36

[5] L Gong, R. Needham and R. Yahalom, "Reasoning about belief in cryptographic protocols," Proceedings of the IEEE Computer Security Symposium on Security and Privacy, May 1990, pp. 234-248

[6] T. Newe and T. Coffey, "Formal Verification logic for hybrid security protocols", International Journal of Computer Systems Science & Engineering, Vol. 18, No. 1, Jan 2003, pp. 17-25.

[7] K. Kim, S. Park, and J. Baek, "Improving Fairness and Privacy of Zhou-Gollmann's FairNon-repudiation Protocol", Proc. of ICPP Workshop on Security (IWSEC), IEEE Computer Society, 1999, pp. 140-145

[8] N. Zhang, Q. Shi "Achieving non-repudiation of receipt", The Computer Journal, Vol.39, No. 10, 1996, pp. 844-853.

[9] Yuyi Ou; Jie Ling; Lu Liang; Xiang Xu "Certified E-mail delivery with offline TTP", Hao Wang, Third International Symposium on Information Assurance and Security, IAS 2007, pp. 29-31

[10] T. Coffey and P. Saidha, "A Logic for Verifying Public-Key Cryptographic Protocols", IEE Proceedings of Computers and Digital Techniques, Vol. 144, No. 1, January 1997, pp. 28-32.

[11] Y. Zhang and V. Varadharajan: "A logic for modelling the dynamics of beliefs in cryptographic protocols". Australasian Computer Science Conference, Gold Coast, Queensland, Australia, February 2001, pp. 215-222.

[12] R. Dojen, T Coffey, "Layered Proving Trees: A novel Approach to the Automation of Logic-Based Security Protocol Verification" ACM Transactions on Information and System Security (TISSEC), Vol. 8, No. 3, August 2005, pp. 287-311 .

[13] R. Dojen, I. Lasc, T. Coffey, R. Gyorodi "Verifying a Key Distribution and Authentication Protocol Using a Logic-Based Proving Engine", RSEE, 2008, pp. 2-3

[14] L. Gong. "A security risk of depending on synchronised clocks". ACM Operating Systems Review, Vol. 26 , No 1, January 1992, , pp. 49-53

[15] D. Veitch and S. Babu and A. P´asztor: "Robust Synchronization of Software Clocks Across the Internet", Proc. Internet Measurement Conference, 2004, pp 50-60

[16] A. Bonnecaze, P. Liardet, A. Gabillon, K. Blibech "Secure time-stamping schemes A distributed point of view", Annals of Telecommunications, Vol. 61, No. 5-6, 2006, pp. 662–681

[17] T.H. Chen, W.B. Lee and G. Horng, "Secure SAS-like password authentication schemes," Computer Standards & Interfaces, Vol.27, 2004, pp. 25-31

[18] Yen-Cheng Chen,Lo-Yao Yeh, "An efficient nonce-based authentication scheme with key agreement," Applied Mathematics and Computation, Vol. 169, 2005, pp. 982-994