具有时限约束的安全协议分析技术研究*)

董荣胜 彭 勋 郭云川 古天龙
(桂林电子工业学院计算机系 桂林541004)

摘 要 本文指出了现有时限责任分析技术中存在的缺陷,提出了一种基于Kailar逻辑的安全协议时限责任分析框架。通过该分析框架对一个具有时限性要求的安全电子投递协议进行分析,发现了协议存在的时限问题,修改了协议并给出了修改后的协议满足时限性要求的证明。

关键词 形式化方法,时限责任,安全协议

On the Analysis of Secure Protocols with Temporal Properties

DONG Rong-Sheng PENG Xun GUO Yun-Chuan GU Tian-Long
(School of Computer Science, Guilin University of Electronic Technology, Guilin 541004)

Abstract In this paper, flaws in the temporal accountability logics are analyzed. A new framework based on Kailar's logic is proposed for the analysis of secure protocols that require temporal accountability. Two temporal flaws are detected by applying this new framework to analyze a time-critical electronic submission protocol, the protocol is modified and the ability to hold temporal accountability of the protocol is also proved.

Keywords Formal methods, Temporal accountability, Security protocol

1 引言

Internet作为通信技术和信息技术的载体得到了前所未有的快速增长,在Internet上出现了日益广泛的经济贸易活动,基于Internet的电子商务应用得到了迅速的发展。电子商务协议是实现电子商务的重要组成部分,需要满足一定的安全特性,对于一类有时间限制的电子商务协议,需要满足的一个特性是时限性。时限性是指协议中各个主体的行为必须在规定的时间范围内完成,否则要求协议能够追究该主体的时限责任。

本文第2节指出了现有时限责任分析技术中存在的缺陷;第3节给出一种安全协议时限责任的分析框架;第4节使用该分析框架对一个电子投递协议进行分析。

2 相关工作及问题分析

在对安全协议进行形式化分析的历程中,逻辑方法已被证明是一种成功的分析方法。1990年,Burrows、Abadi和Needham提出了一个基于知识和信仰的逻辑[1],简称BAN逻辑。BAN逻辑是第一个对安全协议进行验证的逻辑。

1993年,Syverson[2]首先在AT逻辑[7]的基础上增加了时态公式,并能对密码协议的因果一致性进行分析。1996年,Stubblebine和Wright在Syverson工作的基础上在时限协议的分析中加入了同步语义[3],并给出了密钥期限和信任公式可撤销(revocation)的概念。以上工作都是基于BAN逻辑的扩展,不能对时限责任进行分析。

1996年,Kailar引入责任的概念,提出了一种用于分析电子商务协议中可追究性的逻辑框架[4,8]。1998年,Kudo提出了时限责任的问题,并在Kailar逻辑的基础上引入了9个新的逻辑结构(logic construct)和10个新的逻辑规则(logic postulate)[5]。

梁坚等人[6]在对Kudo逻辑进行分析后发现该逻辑第7条逻辑规则没有加入对消息新鲜性的判断,因而不能正确分析发生消息重放时协议各方的责任。在此分析基础上,文[6]对Kailar逻辑进行了新的扩展,增加了4条逻辑结构和2条逻辑规则。

增加的4条逻辑结构是:

1) x At t:在时间 t 发生了 x。x 为任意的公式,如 A Says x At t, A Receives x At t。

2) x TimestampedWith t:对 x 打上时间标记 t。

3) x FreshstampedWith n:对 x 打上新鲜标记 n。在某些协议中,时间标记同时也作为新鲜性的标志。

4) x FreshBefore t:x 在时间 t 之前是新鲜的(包括 t 时刻),即 x 在 t 时刻以前是唯一的。

增加的2条推理规则为:

1)新鲜性规则

A CanProve (x FreshstampedWith n SignedWith K^{-1})
A CanProve (x TimestampedWith t SignedWith K^{-1})
A CanProve (K Authenticates TTP)
A CanProve (TTP IsTrustedOn t)
$\Rightarrow$ A CanProve (x FreshBefore t)

2)时限规则

A Receives (x TimestampedWith t SignedWith K^{-1})
(x in y) SignedWith K_B^{-1}
A CanProve (x FreshBefore t)
A CanProve (K_B Authenticates B)
A CanProve (K Authenticates TTP)
A CanProve (TTP IsTrustedOn t)
$\Rightarrow$ A CanProve (B Says y At t)

文[6]提出的分析技术修正了Kudo逻辑在判断消息新鲜性方面的缺陷,同时带来以下三个新的问题:

第一,新鲜性规则认为消息 x 的新鲜与否取决于时间戳

*)本文得到国家自然科学基金(60243002)资助。

服务器提供的时间戳。然而，时间戳服务器只负责对通过它传输的消息加盖时间戳，以证明通过时间戳服务器的消息是在加盖时间戳的 t 时刻收到的(通信的内容出于保密性要求不能让其它方知道)。因此，协议中消息的新鲜与否应取决于协议中对消息的规定，而不是时间戳服务器的时间戳值。例如有的协议采用临时值或者是序列号来保证协议中消息的新鲜性。

第二，根据时限规则，B 向 A 发送的消息应该是形如：

$\{\{x,t\}K_T^{-1}\}K_B^{-1}$

也即，用户 B 在向用户 A 发送消息时，先将这个消息发送给服务器 T，得到 T 的时间戳签名后，再用自己的私钥进行签名，随后再发送给用户 A。然而，用户 B 可以在得到 T 的时间戳之后，等待一个很长的时间，随后再将这个消息签名发送给 A。因此，这个消息结构不能证明 B 何时发送了这个消息。T 的时间戳签名只能证明 T 是在 t 时刻收到消息 x 的，至于 B 是什么时候再将这个消息发送给用户 A，A 无法得知。

因此，该规则的第二条语句应修改为：

(y in x) SignedWith K_B^{-1}

这样参与者 A 收到的消息的内容是形如：

$\{\{y\}K_B^{-1},t\}K_B^{-1}$

第三，文[6]对时限责任进行分析时，倾向于分析协议能否保证消息的传输过程能够在某个特定的时限内完成。然而协议时限性分析的内容应该是协议的各个参与方在收到请求后能否及时地作出响应。例如，商家在收到用户的购物请求后，能否在规定的时间内作出回应。因此，分析的目标是协议能否保证商家在收到请求的时刻 t_1 开始到做出回应的时刻 t_2 能够在一个使客户满意的时间段内，而不是对从客户开始发送请求的时刻 t_0 到商家收到这个请求的时刻 t_1 的时间段分析。由于 t_0 到 t_1 这个时间段是消息在网络中传输的过程，商家和客户都无法保证消息能在规定的时间段内到达对方，因此，双方都没有必要对此负责。

另外，通过对 Kudo 逻辑的进一步分析，本文认为，即使 Kudo 逻辑的第7条规则加入了对消息新鲜性的判断，这条规则仍有缺陷。

Kudo 逻辑的第7条规则如下：

7)Says At

A CanProve (x GeneratedBefore t)
(y SignedWith K^{-1}) in x
A CanProve (K Authenticates B)
=> A CanProve (B Says y At t)

这条规则描述的是：如果 A 能够证明在 t 时刻或 t 时刻以前接收到了包含有 B 签名的一条消息 y，那么 A 就能够证明 B 是在 t 时刻产生的 y。如果假设在 t 时刻 A 收到了以下的消息：

$\{\{y\}K_B^{-1},t\}K_T^{-1}$

可以看出，这个消息是时间戳服务器 T 在对 B 发送的消息加盖时间戳之后再转发给用户 A 的。那么 A 可以判断：由于只有 B 先发送签名消息，T 才可能对该消息加盖时间戳，因此 B 是在 t 时刻之前发送这个消息的。但是这不足以判断 B 是在 t 时刻发送消息的，因为：

1) B 可以在发送了这个消息之后，在后面的某个时刻再次重发该消息；

2) 如果 B 发送消息到时间戳服务器的通信延迟需要考虑，则不能判断时间戳服务器 T 对消息 y 加盖时间戳的时刻即是 B 发送消息的时刻。

所以，这条规则应该修改为：

7')Says At

A CanProve (x GeneratedBefore t)
(y SignedWith K^{-1}) in x
A CanProve (K Authenticates B)
=> A CanProve (y GeneratedBefore t By B)

可以看出，如果不在第7条规则的基础上增加 B 和时间戳服务器之间通信延迟可以忽略不计的假设，那么即使原有规则加入了对消息新鲜性的判断也不能推导出 B Says y At t。

3 时限责任分析框架

通过对已有时限逻辑的总结，本文在 Kailar 逻辑的基础上扩展了一种新的分析框架。下面先简要介绍 Kailar 逻辑。

3.1 Kailar 逻辑简介

Kailar 逻辑是 Kailar 提出的一种对电子商务责任性进行形式化逻辑证明的方法。下面给出 Kailar 逻辑的基本符号、语法和语义。

A、B：参与协议的各个主体；

K 是密钥，K_a 为主体 A 的公钥，Ka^{-1} 是与 K_a 相对应的 A 的私钥；

m 表示消息。

TTP：可信任第三方(Trusted Third Party，简称 TTP)。

Kailar 逻辑的公式如下：

A CanProve x：对于任何主体 B，A 能够通过执行一系列操作后使得 B 相信公式 x，同时不泄漏任何秘密 $y(y\neq x)$ 给 B，这种证明叫强证明；

A CanProve x to B：表示 A 可以向 B 证明 x，而不泄漏任何其他秘密 $y(y\neq x)$ 给其他主体，这种证明叫弱证明；

A IsTrustedOn x：A 对公式 x 具有管辖权，即任何主体都相信 A 所声明的公式 x 是正确的；

K_a Authenticates A：表示可以通过公钥 K_a 来认证 A 的数字签名；

x in m：表示 x 蕴含在消息 m 中；

A Says x：表示 A 声明公式 x 并对 x 以及所有 x 能够推导出的公式负责。这一个公式隐含有以下推导：A Says (x,y) $\Rightarrow A$ Says x；

A Receives m SignedWith K^{-1}：表示 A 收到了用 K^{-1} 签名的消息 m；

$x\Rightarrow y$：表示 x 蕴含 y，即若 x 成立，则 y 成立；

Kailar 逻辑的基本规则：

1. 连接规则：

$$\frac{A\ \text{CanProve}\ x;A\ \text{Canprove}\ y}{A\ \text{Canprove}(x\wedge y)}$$

如果 A 能够证明公式 x，并且 A 能够证明公式 y，那么 A 就能够证明公式 $x\wedge y$；

2. 推理规则：

$$\frac{A\ \text{CanProve}\ x;x\Rightarrow y}{A\ \text{Canprove}\ y}$$

如果 A 能证明 x，x 蕴涵 y，那么 A 能够证明 y；

3. 信仰规则：

A Believes $x\Leftrightarrow A$ CanProve x to A

A 相信 x，当且仅当 A 能向自己证明 x 成立；

4. 签名规则：

$$\frac{A\ \text{Receives}\ (m\ \text{SignedWith}\ K^{-1});x\ \text{in}\ m;\quad A\ \text{CanProve}\ (K\ \text{Authenticates}\ B)}{A\ \text{CanProve}\ (B\ \text{Says}\ x)}$$

如果 A 收到用私钥 K^{-1} 签名的消息 m，m 中包含了 x，并且 A 能够证明公钥 K 能够验证 B 的身份，那么 A 就能够证明 B 对公式 x 负责；

5. 信任规则：

$$\frac{A\ \text{CanProve}(B\ \text{Says}\ x);A\ \text{CanProve}(B\ \text{Is TrustedOn}\ x)}{A\ \text{CanProve}\ x}$$

如果 A 能够证明 B 声明了公式 x，并且 A 能够证明 B 对 x 具有管辖权，那么 A 能够证明 x。

3.2 扩展后的时限责任分析框架

为了使 Kailar 逻辑能够分析时敏协议的时限责任，有必要增加分析协议参与者敏感行为产生时间的内容。本文引入 BAN 逻辑中的逻辑结构 Fresh 对协议中主体所接收到的消息的新鲜性进行判断，并在 Kailar 逻辑的基础上增加7条新的逻辑结构和4条新的逻辑规则。

新增的7条逻辑结构是：

1) At t：表示事件在时刻 t 发生。如 Receives x At t，Says x At t。

2) Before t：表示事件在时刻 t 之前发生。例如 A Says x Before t To B 表示 A 是在 t 时刻之前发送消息 x 给 B。

3) After t：表示事件在时刻 t 之后发生。例如 A Receives x After t From B 表示 A 是在 t 时刻之后收到来自 B 的消息 x。

4) ResponseDuring$[t_1,t_2]$：表示某个主体在时刻 t_1 到 t_2 的范围内做出了响应。例如 A ResponseDuring$[t_1,t_2]$ To B 表示 A 在时间间隔$[t_1,t_2]$的范围内对 B 的请求做出了响应。

5) TimestampedWith t：表示消息被加盖了时间戳 t。例如 x TimestampedWith t Signed With K_T^{-1} 表示消息 x 被加盖了时间戳 t，随后使用 K_T^{-1} 签名。而 K_T^{-1} 一般是时间戳服务器的签名。

6) Fresh(x)：表示消息 x 是新鲜的。也即，x 在协议的当前运行之前未在任何其它消息中发送过。由于临时值是为了表示消息的新鲜性而产生的，因此对于临时值这种情况通常都是成立的。而临时值通常为时间戳或者是一个仅使用过一次的数字(例如序列号)。

7) CommunicationDelay(A,T)：表示参与者 A 与时间戳服务器 T 之间的通信延迟。这个逻辑结构主要用于对协议运行的初始假设的判断。

逻辑结构 Fresh(x)是从 BAN 逻辑引用过来的，用于对消息新鲜与否的判断。

在给出新增的逻辑规则之前，先给出一条有关时限协议的假设(A 表示协议中的一般主体)P1：CommunicationDelay(A,T)=0，即参与者 A 与时间戳服务器 T 的通信时间可以忽略不计。根据时限协议对于这条假设成立与否的支持，将新增规则中的3条区分为强逻辑规则和弱逻辑规则。

1)确认接收规则

确认接收规则描述了如何判断协议参与者接收某个消息的时刻。

强确认接收规则：

$$\frac{\begin{array}{l}B\ \text{Receives}\ (x\ \text{TimestampedWith}\ t\ \text{SignedWith}\ K_T^{-1})\\ K_T\ \text{Authenticates}\ T\\ T\ \text{Is Trustedon}\ t\\ (y\ \text{in}\ x)\text{SignedWith}\ K_A^{-1}\\ K_A\ \text{Authenticates}\ A\\ \text{CommunicationDelay}(B,T)=0\end{array}}{B\ \text{CanProve}\ (B\ \text{Receives}\ y\ \text{At}\ t\ \text{From}\ A)}$$

其中的 x 指代的是 B 所收到的包括时间戳服务器签名在内的整个消息。例如，假设 B 收到以下消息：

$\{\{\{m\}K_A^{-1},t\}K_T^{-1}\}K_B$

那么可以看出，这个消息是从时间戳服务器 T 转发给 B 的。也即，A 所发送消息内容应该是：

$\{\{m\}K_A^{-1},B\}K_T$

这个消息先被传递到了时间戳服务器 T，T 对消息加盖时间戳之后，再将消息转发给指定的用户 B。因此 B 收到了消息：

$\{\{\{m\}K_A^{-1},t\}K_T^{-1}\}K_B$

规则中的 y 指的是$\{m\}K_A^{-1}$ 部分，x 指代的是 B 所收到的包括时间戳服务器签名在内的整个消息$\{\{m\}K_A^{-1},t\}K^{-1}S_T$。

如果时间戳服务器加盖时间戳的时间可以忽略不计，则有：

$$\frac{\begin{array}{l}B\ \text{Receives}(x\ \text{TimestampedWith}\ t\ \text{SignedWith}\ K_T^{-1}\\ K_T\ \text{Authenticates}\ T\\ T\ \text{Is Trusted On}\ t\end{array}}{B\ \text{CanProve}\ (T\ \text{Says}\ x\ \text{At To}\ B)}$$

也即，如果 B 收到了来自时间戳服务器 T 的消息，而且时间戳服务器加盖时间戳的时间可以忽略不计，那么 B 能够证明：时间戳服务器 T 为消息 x 加盖时间戳的时刻 t，t 也即是消息 x 发送的时刻。

在这种情况下，如果 T 和 B 之间通信的时间间隔可以忽略不计的话，就可以得到：

B CanProve (B Receives y At t From A)

也即：B 能够证明：时间戳服务器为消息 y 加盖时间戳的时刻 t，t 也即是参与者 B 收到消息 y 的时刻。

强确认接收规则的推导中，如果最后一个公式 CommunicationDelay(B,T)=0不成立，也即在参与者 B 与时间戳服务器之间的通信延迟需要考虑的情况下，确认接收规则变为弱确认接收规则。称为弱确认接收规则的原因是：协议不能确切地判定参与者接收某个消息的时刻。

弱确认接收规则：

$$\frac{\begin{array}{l}B\ \text{Receives}(x\ \text{TimestampedWith}\ t\ \text{SignedWith}\ K_T^{-1})\\ K_T\ \text{Authenticates}\ T\\ T\ \text{Is TrustedOn}\ t\\ (y\ \text{in}\ x)\text{SignedWith}\ K_A^{-1}\\ K_A\ \text{Authenticates}\ A\end{array}}{B\ \text{CanProve}(B\ \text{Receives}\ y\ \text{After to From}\ A)}$$

在参与者 B 与时间戳服务器之间的通信延迟需要考虑的情况下，虽然仍可以假设时间戳服务器加盖时间戳的时间可以忽略不计，从而得到：

$$\frac{\begin{array}{l}B\ \text{Receives}(x\ \text{TimestampedWith}\ t\ \text{SignedWith}\ K_T^{-1})\\ K_T\ \text{Authenticates}\ T\\ T\ \text{Is TrustedOn}\ t\end{array}}{B\ \text{CanProve}\ (T\ \text{Says}\ x\ \text{At}\ t\ \text{To}\ B)}$$

但是，由于从 T 发送消息 x 到 B 有一定的延迟，因此协议不能确定 B 收到消息的时刻，而只能推断 B 是在 T 为消息加盖时间戳 t 之后得到的消息，即：

B CanProve (B Receives y After t From A)

2)确认发送规则

确认发送规则描述了如何判断协议参与者发送某个消息的时刻。

强确认发送规则：

$$\frac{\begin{array}{l}B\ \text{Receives}(x\ \text{TimestampedWith}\ t\ \text{SignedWith}\ K_T^{-1})\\ K_T\ \text{Authenticates}\ T\\ T\ \text{Is Trusted On}\ t\\ (y\ \text{in}\ x)\text{SignedWith}\ K_A^{-1}\\ K_A\ \text{Authenticates}\ A\\ \text{Fresh}(y)\end{array}}{}$$

$$\frac{\text{CommunicationDelay}(A,T)=0}{B\ \text{CanProve}(A\ \text{Says}\ y\ \text{At}\ t\ \text{To}\ B)}$$

确认发送规则和确认接收规则之间的区别可以使用图1来解释。

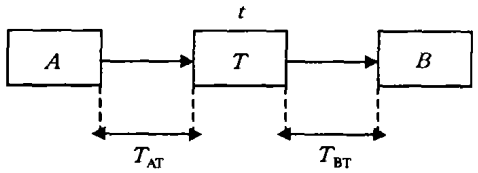

图1 确认发送规则和确认接收规则的区别

如上图所示，T_{AT}表示参与者 A 和时间戳服务器 T 之间的通信时间，T_{BT}表示参与者 B 和时间戳服务器 T 之间的通信时间，而 t 是 T 为 A、B 之间消息通信加盖时间戳的时刻。

确认接收规则是用于使消息的接收者确认消息的接收时刻的，而确认发送规则是用于使消息的接收者确认消息的发送时刻的。通过对两条规则的比较，可以看出：在两条规则中，都假设消息的接收者收到了相同的消息，但做出的判断却不同。这里的关键在于协议中的假设 P1。在确认接收规则中，判断的是 CommunicationDelay$(B,T)=0$，而在确认发送规则中判断的是 CommunicationDelay$(A,T)=0$。

参考图1，直观地，如果参与者 A 和时间戳服务器 T 之间的通信时间 T_{AT}可以忽略不计的话，那么可以判断：时间戳服务器 T 为 A 发送的消息加盖时间戳的时刻 t，t 也即是参与者 A 发送消息的时刻；而如果参与者 B 和时间戳服务器 T 之间的通信时间 T_{BT}可以忽略不计的话，那么可以判断：时间戳服务器 T 为 A 发送的消息加盖时间戳的时刻 t，t 也即是参与者 B 接收到消息的时刻。更一般地，如果 T_{AT}和 T_{BT}都可以忽略不计的话，就相当于现实世界中的面对面交谈了，在这种情况下，发送和接收消息的时刻可以认为是相同的。

确认发送规则和确认接收规则的另一个不同之处在于：前者加入了对消息 y 新鲜与否的判断 Fresh(y)。由于时间戳服务器只负责对 A 和 B 之间通信的消息加盖时间戳，而不能判断消息是否新鲜，发生消息的重放是有可能的。因此，为了确定消息的具体发送时刻，有必要判断消息的内容是否新鲜。另外，消息的新鲜与否和协议规定的消息内容有关，而与时间戳服务器加盖的时间戳无关。

和弱确认接收规则类似的是，如果强确认发送规则的推导中最后一个公式 CommunicationDelay$(A,T)=0$不成立，也即在参与者 A 与时间戳服务器之间的通信延迟需要考虑的情况下，确认发送规则变为弱确认发送规则。称为弱确认发送规则的原因是，协议不能确切地判断参与者发送某个消息的时刻。

弱确认发送规则：

$$\frac{\begin{array}{l}B\ \text{Receives}(x\ \text{TimestampedWith}\ t\ \text{SignedWith}\ K_T^{-1})\\ K_T\ \text{Authenticates}\ T\\ T\ \text{Is Trusted On}\ t\\ (y\ \text{in}\ x)\text{Signed With}\ K_A^{-1}\\ K_A\ \text{Authenticates}\ A\\ \text{Fresh}(y)\end{array}}{B\ \text{CanProve}(A\ \text{Says}\ y\ \text{Before}\ t\ \text{To}\ B)}$$

在参与者 A 与时间戳服务器之间的通信延迟需要考虑的情况下，虽然仍可以假设时间戳服务器加盖时间戳的时间可以忽略不计，从而得到：

$$\frac{\begin{array}{l}B\ \text{Receives}(x\ \text{TimestampedWith}\ t\ \text{SignedWith}\ K_T^{-1})\\ K_T\ \text{Authenticates}\ T\\ T\ \text{Is TrustedOn}\ t\end{array}}{B\ \text{CanProve}(T\ \text{Says}\ x\ \text{At}\ t\ \text{To}\ B)}$$

但是，由于从 A 发送消息 x 到 T 有一定的延迟，因此协议不能确定 A 发送消息的时刻，而只能推断 A 是在 T 为消息加盖时间戳 t 之前发送了消息，即：

B CanProve (A Says y Before t To B)

3)信任传递规则

$$\frac{\begin{array}{l}B\ \text{Receives}\ x_2\ \text{SignedWith}\ K_A^{-1}\\ K_A\ \text{Authenticates}\ A\\ x1\ in\ x2\\ A\ \text{CanProve}(A\ \text{Receives}\ x1\ \text{At}\ t\ \text{From}\ B)\end{array}}{B\ \text{CanProve}\ (A\ \text{Receives}\ x1\ \text{At}\ t\ \text{From}\ B)}$$

根据 Kailar 逻辑中强证明的语义，如果 A CanProve (x)，那么所有的其它参与者都将相信 x 的正确性。这样，如果 B 收到了来自 A 签名的消息，且消息中包含了 x，那么 B 也将相信 x 的正确性。

4)确认时限规则

时限逻辑的分析目标就是要确认协议参与者的行为有无违反时限要求，因此，分析时最重要的就是确定参与者行为的时间间隔。确认时限规则用于判断协议参与者行为的时间间隔。

强确认时限规则(假设 $t_1<t_2$，并假设 x_1是为 B 向 A 发送的请求消息，而 x_2是为 A 向 B 发送的相应的响应消息)：

$$\frac{\begin{array}{l}B\ \text{CanProve}(A\ \text{Receives}\ x1\ \text{At}\ t1\ \text{From}\ B)\\ B\ \text{CanProve}(A\ \text{Says}\ x2\ \text{At}\ t2\ \text{To}\ B)\end{array}}{B\ \text{CanProve}(A\ \text{ResponseDuring}[t1,t2]\text{To}\ B)}$$

这条规则描述的是：如果参与者 B 能够证明参与者 A 是在时刻 t_1接收到来自 B 的请求消息 x_1的，并且 B 还能够证明参与者 A 是在时刻 t_2发送响应 x_2给自己的，那么 B 能够证明：A 是在时间间隔 t_1到 t_2范围内响应来自 B 的消息 x_1的(如图2所示)。

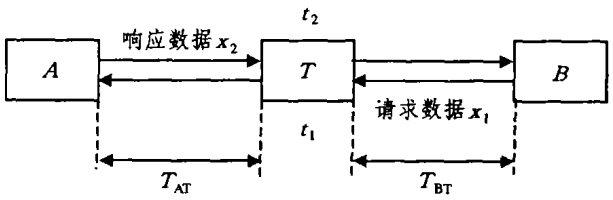

图2 确认时限规则

这个规则被称为强确认时限规则的原因是 B 能够证明 A 对 B 的请求作出响应的确切的时间范围(时间间隔为闭区间)。实现参与者之间相互的强确认时限的一个方法是：在参与者 A 和 B 两方各采用一个时间戳服务器 T_1和 T_2，并对 A 和 B 发送与接收消息的时刻都打上时间戳标记。这样的通信模型可见图3。相反，如果只要有一个时刻 B 不能确切判断(如 A 接收请求的时刻，或者是 A 发送响应的时刻)，那么确认时限规则就变为弱确认时限规则，同时 A 的响应时间间隔变为至少有一边是开区间的。

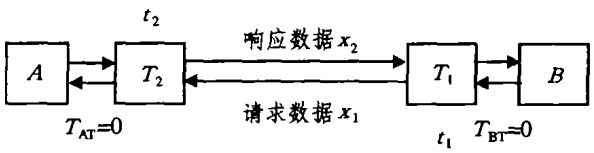

图3 实现参与方之间相互强确认时限的一种方法

根据 B 能否证明 A 接收请求和 A 发送相应的响应的确切时刻，可以划分出3种弱确认时限规则，在这里本文只列举一种情况：B 既不能确定 A 接收到请求的时刻，也不能确定

A 发送响应的时刻。在这种情况下，弱确认时限规则为(假设 $t_1<t_2$，并假设 x_1 是为 B 向 A 发送的请求消息，而 x_2 是为 A 向 B 发送的相应的响应消息)：

$$\frac{B\ \text{CanProve}(A\ \text{Receives}\ x1\ \text{After}\ t1\ \text{From}\ B)\quad B\ \text{CanProve}(A\ \text{Says}\ x2\ \text{Before}\ t2\ \text{To}\ B)}{B\ \text{CanProve}\ (A\ \text{ResponseDuring}(t1,t2)\text{To}\ B)}$$

4 对一个电子投递协议的分析

Kudo 认为最典型的具有时限约束的应用是申请书的电子投递，该投递只在特定的时限内才有效，为此给出了一个安全电子投递协议。协议有3个参与方，分别为申请者 A，投递的接收者 B 以及一个受信任的时间戳服务器 T。

该协议要满足以下要求：

R1) 接收者只能在允许的期限内接收投递；

R2) 申请者必须在允许的期限内发送投递数据；

R3) 接收者不能够伪造、修改或者是删除收到的任何投递；

R4) 对于在允许的期限内到达的投递，接收者必须接受。

4.1 协议的假设

N1)时间戳服务的提供者是被各方所信任的，能提供当前的精确时间并产生正确的时间戳；

N2)申请者和接收者的系统时钟是不可信的；

N3)接收者要公平地提供投递服务给所有的申请者；

N4)接收者和时间戳服务器的计算资源是无限的，而且它们通信或者是执行协议的时间可以忽略不计；

N5)数字签名算法基于公钥密码学，是不可修改的；

N6)协议中没有使用拒绝服务攻击的敌对方；

N7)参与者之间的通信信道是安全的。

4.2 协议的过程

投递协议包含4个阶段：预备阶段、开放阶段、投递阶段和关闭阶段。在预备阶段中，B 请求 T 通知投递的开放时间和关闭时间。当投递的开放时刻到来时，T 通知接收者进入开放阶段。在投递阶段，申请者发送申请数据给接收者 B 并收到了 B 对申请的确认时间戳。在关闭阶段中，B 向 T 请求一个关闭时间戳，随后将其发送给申请者。下面是协议的具体过程(协议省略了用接收方的公钥进行加密的过程)。

4.2.1 预备阶段

1) $B\text{->}T$: NR, $\{T_i, T_c\}K_B^{-1}$
2) $T\text{->}B$: NA, $\{\{T_i, T_c\}K_B^{-1}\}K_T^{-1}$

NR 为通知请求，NA 为请求确认。在消息1中，B 请求 T 在时刻 T_i 和 T_c 到来时通知 B。T 接受 B 的请求并向 B 发送请求确认。

4.2.2 开放阶段

3) $T\text{->}B$: T_i
4) $B\text{->}T$: $\{\text{desc}\}K_B^{-1}$
5) $T\text{->}B$: $\{T_i, \{\text{desc}\}K_B^{-1}\}K_B^{-1}$

T_i 时刻到来时，T 发送消息3给 B 并通知 B 投递时间开始。其中 desc 是投递描述，内容为 $\{T_i, T_c\}$。T 对这个消息加盖时间戳以便使申请者能够在消息8中知道投递的期限。

4.2.3 投递阶段

6) $A\text{->}B$: id
7) $B\text{->}A$: $\{T_i, \{\text{desc}\}K_B^{-1}\}K_T^{-1}$
8) $A\text{->}B$: $\{X_i, A, B, \{T_i, \{\text{desc}\}K_B^{-1}\}K_T^{-1}\}K_A^{-1}$
9) $B\text{->}T$: $\{Y_i\}K_B^{-1}$
10) $T\text{->}B$: $\{T_1, \{Y_i\}K_B^{-1}\}K_T^{-1}$
11) $B\text{->}A$: $\{T_1, \{Y_i\}K_B^{-1}\}K_T^{-1}$

在投递阶段，申请者 A 向 B 发送一个投递请求的序号。B 在收到这个消息后向 A 转发消息5中 T 产生的消息，这样 A 获得了投递的期限。在消息8中，A 向 B 发送电子投递，消息的内容包含了申请数据 X_i、发送方、接收方、时间戳以及 A 的签名，其中 i 表示 A 的第 i 次投递。显然，在消息8中包含 A 收到的投递期限，这样可以证明 A 是在 T_i 之后发送的申请请求。

消息9中的 Y_i 是消息8中的内容。接收者收到申请者的投递后，若没有超过关闭时间，B 将请求 T 为此消息加盖时间戳；若超过期限，B 将发送消息15，而不会发送消息9和11。

若 T 收到消息9，T 将为消息加盖时间戳然后返回给 B。B 随后向 A 转发消息10。若 A 收到该消息，就能验证 B 收到了消息8。

4.2.4 关闭阶段(Closing Step)

12) $T\text{->}B$: T_c
13) $B\text{->}T$: $\{Z\}K_B^{-1}$
14) $T\text{->}B$: $\{T_c, \{Z\}K_B^{-1}\}K_T^{-1}$
15) $B\text{->}A$: $\{T_c, \{Z\}K_B^{-1}\}K_T^{-1}$

其中 $Z=(h(Y_1), h(Y_2), h(Y_3), \cdots, h(Y_n))$。

关闭时刻 T_c 到来时，T 发送消息12通知 B 关闭时刻到达。B 随后向 T 发送一个所有成功收到投递的哈希值计算结果。T 对 B 发送的结果加盖时间戳然后返回给 B。随后 B 向所有申请者发送此消息。A 在收到该消息后，可以检查自己是否在成功提交的用户列表中。

4.3 协议存在的问题

Kudo 提出的电子投递协议存在两个问题：

问题1：在消息8中，A 发送的消息没有表示新鲜与否的信息。消息中 T 的签名是对起止时间的认证，接收者 B 和时间戳服务器 T 都不能判断消息8的新鲜与否。如果在投递过程中有对消息的重传，那么在 A 收到的消息11和消息15中将包含有比前一个消息要晚的时刻 T_2 的时间标记。在此情况下，协议的各方都不能判断接收方 B 接收到消息的时刻，使得各方的时限责任难以追究。

问题2：如果 B 收到 A 发送的投递数据的时刻和投递的关闭时刻很接近，那么在出于敌意的情况下 B 可能会将 A 的申请数据延迟到关闭投递之后，使得 A 无法完成投递。由于 A 无法证明 B 收到申请的时刻，因此 A 无法证实到底是出于网络问题而造成的投递延迟，还是 B 有意造成的延迟，从而无法追究 B 的时限责任。

4.4 协议的改进及时限责任分析

使用本文改进后的分析框架可以方便地验证协议存在以上两个问题以及产生这两个问题的原因，下面给出修改后的电子投递协议的投递阶段的描述，并证明修改后的协议能提供对接收者 B 的时限责任的追究。

修改后电子投递协议中的投递阶段如下：

6) $A\text{->}B$: Submission Request
7) $B\text{->}A$: $\{T_i, \{\text{desc}\}K_B^{-1}\}K_T^{-1}$, SID
8) $A\text{->}T$: $\{\text{SID}, S_i, X_i, A, B, \{T_i, \{\text{desc}\}K_B^{-1}\}K_T^{-1}\}K_A^{-1}, B$
9) $T\text{->}B$: $\{T_1, \{\text{SID}, S_i, X_i, A, B, \{T_i, \{\text{desc}\}K_B^{-1}\}K_T^{-1}\}K_A^{-1}\}K_T^{-1}$
10) $B\text{->}T$: $\{Y_i\}K_B^{-1}$
11) $T\text{->}A$: $\{T_2, \{Y_i\}K_B^{-1}\}K_T^{-1}$
12) $T\text{->}B$: $\{T_2, \{Y_i\}K_B^{-1}\}K_T^{-1}$

在新的投递阶段中，申请者 A 首先向接收者 B 发送一个投递请求。B 在消息7中将一个投递序号 SID 和投递期限发送给 A。在投递阶段的后续过程中，申请者 A 将使用这个 SID 来发送数据，并使得每次的投递数据 X_i 对应一个投递次数

S_i。这样，B 接收到 A 发送的投递数据后，只要检查 SID 和 S_i 是否有重复即可，而不必检查整个投递的数据 X，是否重复。如果 B 确认了投递，就会向服务器 T 发送消息10请求服务器加盖 B 的确认时间戳。最后，T 向 A、B 返回加盖时间戳后的结果。

修改前后投递阶段各方的通信过程可表示成图4和图5。比较两图，容易看出修改前后的区别。

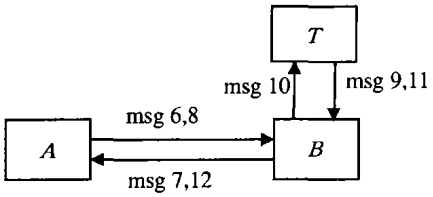

图4 修改前投递阶段的通信过程

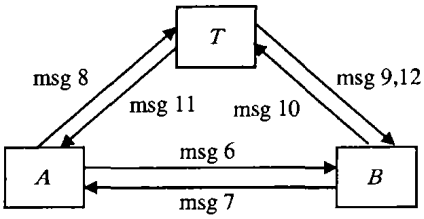

图5 修改后投递阶段的通信过程

由于本文仅分析协议中接收者 B 的时限责任，而对 B 的时限要求是：要求 B 能够在规定的时间间隔内处理完申请者的请求，因此得到协议的预期目标是：

A CanProve (B ResponseDuring$[T_1,T_2]$ To A)这个目标的含义是：申请者 A 能够证明 B 在时间间隔$[T_1,T_2]$内响应了自己的请求。

下面使用改进后的分析框架来分析这个协议，使用时限逻辑对协议进行分析的一般步骤是：

1) 建立协议要满足的目标集合 G；

2) 建立协议的逻辑模型，也即将协议的语句转换为逻辑公式；

3) 建立协议的初始化假设集合 A；

4) 运用初始假设和逻辑规则对协议进行推导和分析。

4.4.1 *协议要满足的目标集合* 根据上述的分析，修改后协议的预期目标是：

G1：A CanProve (B ResponseDuring$[T_1,T_2]$ To A)

4.4.2 *协议的逻辑模型* 协议的投递阶段可以被描述如下(本文将各个非 T 处理的消息字段省略为 $M6$，$M7\cdots M12$，以指代修改后投递阶段的各个消息)：

(6) B Receives $M6$;
(7) A Receives $M7$;
(8) T Receives ($M8$ SignedWith K_A^{-1})
(9) B Receives $M8$ TimestampedWith T_1 SignedWith K_T^{-1}
(10) T Receives ($M10$ SignedWith K_B^{-1})
(11) A Receives ($M10$ SignedWith K_B^{-1}) TimestampedWith T_2 SignedWith K_T^{-1}
(12) B Receives ($M10$ SignedWith K_B^{-1}) TimestampedWith T_2 SignedWith K_T^{-1}

4.4.3 *协议的初始假设条件*

A1：A,B CanProve (T IsTrustedOn T_1,T_2,T_i,T_c)
A2：K_B Authenticates B
A3：K_A Authenticates A
A4：K_T Authenticates T
A5：CommunicationDelay(B,T)=0

4.4.4 *对协议目标的分析过程*

由

B Receives $M8$ TimestampedWith T_1 SignedWith K_T^{-1}
K_T Authenticates T
T IsTrustedOn T_1
$M8$ SignedWith K_A^{-1}
K_A Authenticates A
CommunicationDelay(B,T)=0 {确认接收规则}
=> B CanProve (B Receives $M8$ At T_1 From A)

由

A Receives ($M10$ SignedWith K_B^{-1}) TimestampedWith T_2 SignedWith K_T^{-1}
$M8$ in $M10$
B CanProve (B Receives $M8$ At T_1 From A)
K_B Authenticates B {信任传递规则}
=> A CanProve (B Receives $M8$ At T_1 From A)···①

由

A Receives ($M10$ SignedWith K_B^{-1}) TimestampedWith T_2 SignedWith K_T^{-1}
K_T Authenticates T
T IsTrustedOn T_1
$M10$ SignedWith K_B^{-1}
K_B Authenticates B
(SID,S_i) in $M10$
Fresh(SID,S_i)
CommunicationDelay(B,T)=0 {确认发送规则}
=> A CanProve (B Says $M10$ At T_2) ···②

由①和②，根据确认时限规则，可得到：
A CanProve (B ResponseDuring$[T_1,T_2]$ To A)。

经过上述分析，证明了改进后的协议能够分析参与者 B 的时限责任。另外还可以看到，在推导 A CanProve (B Says $M10$ At T_2)时，判断消息 $M10$新鲜与否的依据是消息 $M10$中的投递序号 SID 和投递次数 S_i 的组合值必须是新鲜的，也即(SID,S_i)不能够在 T_2时刻以前出现在其它任何消息中。对(SID,S_i)值的检查可由接收者 B 进行，如果 B 收到了重复的(SID,S_i)值，则认为是有人重发了该数据并丢弃该次的投递数据。

参 考 文 献

1 Burrows M, Abadi M, Needham R M. A logic of authentication: [Report. 39]. Palo Alto, CA: DEC System Research Center, 1989

2 Syverson P. Adding time to a logic of authentication [A]. In: Proc. of the 1st ACM Conf. on Computer and Communications Security[C]. Fair-fax: ACM, 1993. 97～101

3 Stubblebine S G, Wright R N. An authentication logic supporting synchronization, revocation and recency. In: 3rd ACM Conf. on Computer and Communications Security, Mar. 1996. 95～105

4 Kailar R. Accountability in electronic commerce protocols [J]. IEEE Trans on Software Engineering, 1996, 22(5): 313～328

5 Kudo M. Electronic submission protocol based on temporal accountability[A]. In: Proc. of 14th Annual Computer Security Application Conf. [C]. Phoenix: ACSA, 1998. 353～363

6 梁坚，敖青云，尤晋元. 安全协议的时限责任分析. 电子学报，2002，30(10)：1450～1454

7 Abadi M, Tuttle M R. A semantics for a logic of authentication [A]. In: Proc. of the 10th Annual ACM Symposium on Principles of Distributed Computing[C]. 1991. 201～216

8 郭云川，古天龙，董荣胜，蔡国永. Kailar 逻辑缺陷的进一步讨论. 计算机工程与应用，2003，39(17)：77～79