Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 212 (2008) 103–118
www.elsevier.com/locate/entcs

Modeling and Verifying Time Sensitive Security Protocols with Constraints*

Ti Zhou[a,1] Mengjun Li [a,2] Zhoujun Li [b,3] Huowang Chen [a,4]

[a] *School of Computer Science, National University of Defence Technology, Changsha, China*

[b] *School of Computer Science and Engineering, BeiHang University, Beijing, China*

Abstract

This paper researches the characteristic of time sensitive protocols and presents a method with simple operations to verify protocols with time stamps and avoid false attacks. Firstly, an extension of π calculus is given to model a time sensitive security protocol. And then, by appending linear arithmetic constraints to the Horn logic model, the extended Horn logic model of security protocols and the modified-version verification method with time constraints are represented. All operations and the strategy of verification are defined for our constraints system. Thirdly, a method is given to determine whether the constraints has a solution or not. Finally, as a result of an experiment, Denning-Sacco protocol with time stamps is verified. The experiment shows that our approach is an innovative and effective method on verifying time sensitive security protocols.

Keywords: time sensitive; security protocol; constraint; formal verification

1 Introduction

In the past two decades, more and more protocols are used for tasks such as key distributions, identity authentications, e-commerce transactions etc. However, security protocols do not always achieve their objectives because of the interleaving run of its infinite sessions and the attacker's deliberate demolishment. To avoid the replay and delay, freshness, counters and time stamps are used to mark the freshness of messages in protocols [6]. Most sophisticate protocols are designed to fix these problems by the use of time stamps which are numbers marking a specific instance of time. The use of time stamps can be described as follows: the sender issues a fresh

* Supported by the National Natural Science Foundation of China under Grant No. 60473057, 90604007, 60703075, 90718017, the National High Technology Research and Development Program of China No. 2007AA010301, and the Research Fund for the Doctoral Program of Higher Education No. 20070006055.

[1] Email: tzhou@nudt.edu.cn

[2] Email: mengjun_li1975@yahoo.com.cn

[3] Email: lizj@buaa.edu.cn

[4] Email: hwchen@nudt.edu.cn

doi:10.1016/j.entcs.2008.04.056

message with the time stamp that marks its time of issue; then, the receiver checks whether the time stamp has not expired to establish the validity of the message. For example, Needham-Schroeder protocol with conventional keys [21], which is the best known of all security protocols, is described as Table 1. The main problem

①	A→S:	A,B,N_A
②	S→A:	$\{N_A, B, Kab,$ $\{Kab, A\}Kbs\}Kas$
③	A→B:	$\{Kab, A\}Kbs$
④	B→A:	$\{N_B\}Kab$
⑤	A→B:	$\{N_B - 1\}Kab$

Table 1
The formal description of Needham-Schroeder protocol

(3)	I(A)→ B:	$\{Kab', A\}Kbs$
(4)	B→ I(A):	$\{N_B\}Kab'$
(5)	I(A)→ B:	$\{N_B - 1\}Kab'$

Table 2
The formal description of freshness attack of NS protocol

①	A→S:	A, B
②	S→A:	$\{B, Kab, T_A, \{Kab,$ $T_A, A\}Kbs\}Kas$
③	A→B:	$\{Kab, T_A, A\}Kbs$
④	B→A:	$\{N_B\}Kab$
⑤	A→B:	$\{N_B - 1\}Kab$

Table 3
The formal description of Denning-Sacco protocol

with this protocol is that B has no way of ensuring that the message ③ is fresh. An intruder can compromise the session key and then replay the appropriate message ③ to B and then complete the protocol like Table 2. This attack is available because the session key is generated randomly by server and is only used in a session, so it's not sophisticate. If it expires, the intruder will hold it. Denning and Sacco suggested fixing the freshness flaw in Needham-Schroeder protocol(NS for short) above by the use of time stamps in [10]. Table 3 is Denning-Sacco protocol(DS for short) which is the improvement version of Needham-Schroeder protocol.

However, there will be lots of false attacks found by most current methods, because they often treat time stamps as freshness or neglect them. By removing the complexity of time, they will have verified the original protocol if they can verify the simplified protocol. They have to check each attack by themselves to make sure the protocol is insecurity. For example, if DS is verified by them, there is a false attack like table 2. In our model, the attack in NS doesn't exist in DS, because the constraints in the sequence of the attack are unsatisfiable in DS. Therefore, our method could avoid false attacks when we verify time sensitive security protocols.

Bruno Blanchet and Martin Abadi have proposed a security protocol model based on Horn logic in [1,3,4]. This model is derived from the linear logic model by abstract interpretation, program transformation and Skolemizing formulae with existential quantifiers. Based on this model, they also present an efficient method which fits for verifying interleaving runs of the security protocol's infinite sessions and terminates for many security protocols. The linear logic model is a state-transition system model of security protocols, and the Horn logic model is an abstract model. [16] proposes an extended Horn logic model for security protocols, and gives the modified-version verification method to construct counter-examples automatically[25]. The counter-examples are represented in standard notation, which is more elegant and intelligible than the representation form based on traces. Based on these theories, [17] develops an efficient verifier SPVT(Security Protocol Verifying Tool). Some classic protocols are verified correctly by SPVT, such as a series of Needham-Schroeder authentication protocols(security and authentication), Yahalom protocol(authentication), and so on.

For the relation between times is the powerful one to determine whether mes-

sages are valid or not, we must research the order of these time stamps. We need use constraints of time variables and clocks to build this relation. In fact, time sensitive security protocols can be naturally specified by constraints programming. Symbolic operation will also be the best way to verify time sensitive security protocols. On basis of [1,3,4,16,17,25] and the linear arithmetic constraints, we propose a constraints-based method for modeling, analyzing and verifying time sensitive protocols as follows: we investigate the characteristics of time stamps, and then, add time factors to the process calculus model and Horn logic model, and present in which conditions time constraints could be solved; since time constraints are made up with linear equations and inequations, we can also work out whether time constraints are satisfiable by using the algorithm for constraints resolution in linear programming. In the result, we strengthen the verification capability of the Horn logic model, and keep its effectiveness on verification. This method can verify security protocols with or without time stamps and report a readable counterexample when there is an attack. This paper will discuss the model and verification of time sensitive protocols. The constraints system is discussed in detail in [18].

There have been some other researches on time sensitive security protocols such as [2,5,7,9,12,13,14,15,19,20,24]. [2] verifies some time-dependent protocols by inductive approach. Though it uses this method to verify Kerberos IV, the application of this method is limited due to the simplified time model. It is very important to make the time model more naturally. [5] presents a symbolic decision procedure for time-sensitive cryptographic protocols with time stamps. It uses logic formulae to describe symbolic constraints. However, it doesn't associate time values to short term keys, that is, it can't verify DS protocol. [9] used MSR to verify security protocols with time stamps. Its verification uses simple logic reductions, and the symbolic exploration method can be used to verify time-dependent secrecy and authentication properties, but it needs to compute symbolic reachability graph. [12] performs a semi-automated analysis of a CSP model (with event-based time) of WMF. The description of time in CSP looks like time in real world, but it needs PVS to find an invariants property, and it can't give an answer when computing does not terminate. In [19], the user specifies the protocol using a more abstract notation, and Casper compiles this into CSP code, suitable for checking using FDR. The user has to define a finite integer set as the range of time stamps, so time stamps are discrete, and Casper can just check a finite number of sessions. [14] simplifies time sensitive protocols into ones without time stamps. It won't lose attacks, but it will find false attacks for simplification versions. In our method, the range of time stamps is in a continuous interval of real number without no the upper bound, and this method can verify protocols with an unbounded number of sessions. Last but not least, constraints assure there is few of false attacks generated by time stamps. [24] has described a temporal logic to reason about authentication protocols and has treated several examples in earlier 1990s. Different people may use this logic to present the same property or protocol in different ways, and automatic reasoning in temporal logic is very difficult, so it's very hard to verify time sensitive protocols automatically in this model. There are many other papers on timeout

and retransmission. R.Corin concerns how to model and verify contiguous time by time automata, and studies attacks in timeout and retransmission in [7]. Gorrieri studies a real-time process algebra presented for the analysis of time-dependent properties, and focuses on composite results in [13]. Jeremy Y.Lee describes an algebra for modeling the real-time aspect of systems in a mobile environment, which is the extended algebra by introducing a timeout operator in [15]. [20] studies a compositional denotational model for Timed CSP, and it focuses on timeout and retransmission, but not in the characteristic of time stamps. However, those methods all cannot construct counter-examples automatically.

The paper is organized as follows. In Section 2, we introduce the process calculus used to describe security protocols formally and naturally. In Section 3, we discuss the extended model on Horn logic with time constraints, translation of a protocol from the process calculus to the Horn logic, and verification on the Horn logic with time constraints. In Section 4, we discuss the characteristic of constraints in our model and give a method to determine if there is a solution for the constraints. In Section 5, we analyze Denning-Sacco protocol in detail. Section 6 concludes the future work.

2 Protocol Modeling in the Process Calculus

Security protocols are distributed parallel programs. They can be described as the parallel composition of multiple role-processes using π-like calculus. We propose a π-like calculus for modeling security protocol, which is Applied-π[23] calculus added with some time events and is described in Table 4.

$M, N ::=$	**Terms**
x, y, z	variable
a, b, c, k	Name
$f(M_1, \cdots, M_n)$	Constructor
$P, Q ::=$	**Processes**
$\bar{c} < M > .P$	Output
$c(x).P$	Input
0	Nil
$P \mid Q$	Parallel
$!P$	Replication
$(\nu a)P$	Restriction
let $x = g(M_1, \cdots, M_n)$ in P	Destruction
if $M = N$ then P	Condition
$begin(M, M').P$	Begin event
$end(M, M').P$	End event
$Check(t).P$	Check event
$Mark(t).P$	Mark event
Set $d = int$ in P	Set event
$Fly(d).P$	Fly event

Table 4
The syntax of the process calculus

processA$\triangleq$
Set $d = d1$ in $c(xB).begin(Bparam, xB).\bar{c} < (host(Kas), xB) > .c(X)$. let $\{XB, XKab, XT1, XTicket\} = decrypt(X, Kas)$ in if $xB{=}XB$ then $Check(XT1).\bar{c} < XTicket > .c(Y)$. let $XN_B = decrypt(Y, XKab)$ in $Fly(d1 + d2).\bar{c} < \{XN_B - 1\}XKab >.0$

processS$\triangleq$
$c(XAB)$.if $XAB = (x, y)$ then if $x = host(Kxs)$ then if $y = host(Kys)$ then $Mark(T1).\bar{c} < \{y, Kab(T1), T1, \{Kab(T1), T1, x\}Kys\}Kxs >.0$

processB$\triangleq$
Set $d = d1 + d2$ in $c(XTicket)$. let $\{XKab, XT1, xA\} = decrypt(XTicket, Kbs)$ in $Check(XT1)$. $(\nu N_B)\bar{c} < \{N_B(XT1)\}XKab >$. $c(xBack)$. let $xN = decrypt(xBack, Kbs)$ in if $xN{=}N_B(XT1) - 1$ then $Fly(d1 + d2).end(Bparam, B).0$

DS$\triangleq$
$(\nu Kas)(\nu Kbs)\bar{c} < host(Kas) > .\bar{c} < host(Kbs) > .$
$((!processA)|(!processS)|(!processB))$

Table 5
The Description Model of DS protocol

Our π-like calculus is similar to the calculus in [4], and also has two parts:

terms(data) and processes(programs). The identifiers x, y, z, and similar ones range over variables, and a, b, c, k range over names. f is a constructor which is used to build terms. The processes are defined similarly to [4], but we add to it four time events: **Check**, **Mark**, **Set**, **Fly**. These events are used to deal with time stamps. **Check** event is used to check whether the time stamp is valid or not. The parameter d represents the lifetime of a message, which is the most allowable network delay. This event appears when the process receives a message with time stamps and need to check its freshness. **Mark** event is used to mark t as the current clock **now** when the sender wants to issue a fresh message with the time of issue, that is, t equals to the current value of **now**. **Set** event is used to assign a value of lifetime to d. **Fly** event is used to declare time rolling by, that is, the current clock **now** passes a period d. Using our π-like calculus, Denning-Sacco protocol can be described as the process DS in Table 5.

The identifier A is short for $host(Kas)$ in this paper, and $decrypt(X, K)$ returns plain text which is encrypted to X by a key K. Notes that $N_B(XT1) - 1$ is a term, but $d1 + d2$ is an algebraic expression. [4] defined secrecy and authenticity in process calculus, and specify the theory. We use this calculus to model the description of protocols, but we use Horn logic to verify them, so the following paper will discuss the verification of Horn logic with constraints in detail.

3 Modeling protocol with time constraints in Horn logic

3.1 Syntax

This paper is interested in verification of security protocols with time stamps, and discusses how to verify time sensitive security protocols in the model with constraints on Horn logic. So we are interested in how to consolidate the protocol model on Horn logic with time constraints so as to model and verify time sensitive security protocols. For the sake of simplicity, this paper will not introduce the verification on Horn logic in [1,3,4].

We first introduce special terms to represent time variables, and time functions. The syntax of the Horn logic model with time constraints is described in Table 6.

The Horn logic uses these predicates: *attacker*, *begin*, and *end*. The fact $attacker(M)$ means that the attacker may have M, $begin(M, N)$ that the event **begin** has been executed with a parameter corresponding to M and environment N, and $end(M, N)$ that **end** has been executed in session list N with a parameter corresponding to M. If the constraint is **true** in a rule, then the rule will ignore this constraint, that is, $H_1 \wedge \cdots \wedge H_n \rightarrow F$ means $H_1 \wedge \cdots \wedge H_n \rightarrow F : true$. For convenience, we define a global clock **now** which is a special place-holder and represents current time. The sender overprints time stamps refer to **now**. When the receiver receives this message, he will check time stamps with the current time. If time stamps have not expired, he will believe the message is still valid. The parameters representing network delays can be assigned in the beginning. For convenience, let $\{M_1, M_2, \cdots, M_n\}$ be a combination of terms $M_1, M_2, \cdots, M_n$.

Term	M,N,Mes::=
time variable	$t_1, t_2, \cdots,$
time function	$tf(t_1, \cdots t_n)$ where $t_1, \cdots, t_n$ are time variables or time functions; if $n = 0$, then tf is a time constant.
variable	x, y, z
name	$a[M_1, \cdots, M_n]$,i,j
function	$f(M_1, \cdots, M_n)$
Atom, Fact	F,C::=
attacker predicate	$attacker(M)$
begin predicate	$begin(M, N)$
end predicate	$end(M, N)$
Constraint	C::=
	$f(x_1, \cdots, x_n)\#g(y_1, \cdots, y_m)\|C_1, C_2$, where f and g are n-ary and m-ary functions which return linear combination of these variables, $\# \in \{<, =, \leq\}$.
Rule	R,R′::=
logic rule	$F_1 \wedge \cdots \wedge F_n \rightarrow F : C$ where C is constraint.

Table 6
The syntax of the model with constraints on Horn logic

3.2 Translation to Horn logic

The model of roles in security protocols is a group of logic rules. In logic rules, encrypted data is represented as term encrypt(x,y) abbreviated to {x}y , where y is a term like Kas denoted a key shared between a and s, and x is a term. d is a parameter representing network delay. If we want to do anything with some rule, we must be sure that its constraints can be satisfiable.

Rules for the attacker The intruder's Dolev-Yao model [11] is very popular in analysis and verification of security protocols. In this model, the intruder can control the network wholly, so he can intercept, retransmit, and fake messages transmitted over open network. He can read any message and block further transmission, decompose a message into parts and remember them, generate fresh data as needed, and compose a new message from known data and send. Besides the classic rules of Dolev-Yao model in [1,3,4], we also add the next rule for time sensitive protocols:

$$\rightarrow attacker(k(t)) : now > \Delta + t$$

This new rule means that the lifetime of the session key k is Δ(Δ is a big time constant and $k(t)$ means that k is generated at the time clock t), and session keys can only be leaked by accident when they have expired.

Rules for the protocol The honest roles are described by the process calculus in Section 2. The translation $[\![P]\!]\rho hC$ of a process P is a set of rules, where the environment ρ is a sequence of mappings $x \mapsto p$ and $a \mapsto p$ from (time) variables and names to patterns, h is a sequence of facts of the form $attacker(M)$ and $begin(M, M')$, and C is a list of constraints. The empty list is denoted by $\emptyset$, with the concatenation of a constraint $aCons$ to the list C is denoted by $C \cup \{aCons\}$. $exp_1 < exp_2 < exp_3$ is short for two constraints: $exp_1 < exp_2$ and $exp_2 < exp_3$. The concatenation of a mapping $x \mapsto M$ to ρ is denoted by $\rho[x \mapsto M]$, where x is a name or a variable. Based on the abstracting rules in [4], we give abstracting rules for our process calculus as follows:

(1) $[\![\mathbf{0}]\!]\rho hC = \emptyset$;

(2) $[\![\mathbf{P}|\mathbf{Q}]\!]\rho hC = [\![\mathbf{P}]\!]\rho hC \cup [\![Q]\!]\rho hC$;

(3) $[\![!\mathbf{P}]\!]\rho hC = [\![\mathbf{P}]\!]\rho[i \mapsto i]hC$, where i is a new variable(session identifier)

(4) $[\![\nu(a)P]\!]\rho hC = [\![\mathbf{P}]\!]\rho[a \mapsto a[\rho(V_0), \rho(V_s)]]hC$, where V_0 is the tuple made up with input variables, V_s is the set of session identifiers, and a becomes a new function symbol;

(5) $[\![c(x).\mathbf{P}]\!]\rho hC = [\![\mathbf{P}]\!]\rho[x \mapsto x](h \wedge \mathbf{attacker}(x))C$;

(6) $[\![\bar{c} < M > .\mathbf{P}]\!]\rho hC = [\![\mathbf{P}]\!]\rho hC \cup \{h \rightarrow \mathbf{attacker}(\rho(M)) : (\rho(C))\})$;

(7) $[\![\mathbf{let}\ x = g(M_1, \cdots, M_n)\ \mathbf{in\ P\ else\ Q}]\!]\rho hC = \bigcup\{[\![\mathbf{P}]\!](\sigma\rho)[x \mapsto \sigma' p'](\sigma h)(\sigma C) | g(p_1, \cdots, p_n) \rightarrow p'$, where, the pair (σ, σ') is a most general unifier one, such that $\sigma\rho(M_1) = \sigma'(p_1), \cdots, \sigma\rho(M_n) = \sigma'(p_n)\} \cup [\![\mathbf{Q}]\!]\rho hC$;

(8) $[\![\mathbf{Begin}(M).\mathbf{P}]\!]\rho hC = [\![\mathbf{P}]\!]\rho(h \wedge \mathbf{begin}(\rho|_{(V_o \cup V_s)}, \rho(M)))C$;

(9) $[\![\mathbf{End}(M).\mathbf{P}]\!]\rho hC = [\![\mathbf{P}]\!]\rho hC \cup \{h \rightarrow \mathbf{end}(\rho(V_s), \rho(M)) : (\rho(C))\}$;

(10) $[\![\mathbf{Check}(t).\mathbf{P}]\!]\rho hC = [\![\mathbf{P}]\!]\rho h(C[now \mapsto old\_now] \cup \{now - d < t < now\})$, where old_now is a new time variable;

(11) $[\![\mathbf{Mark}(t).\mathbf{P}]\!]\rho hC = [\![\mathbf{P}]\!]\rho h(C \cup \{t = now\})$;

(12) $[\![\mathbf{Set}\ d = int\ \mathbf{in\ P}]\!]\rho hC = [\![\mathbf{P}]\!](\rho[d \mapsto int])hC$;

(13) $[\![\mathbf{Fly}(d).\mathbf{P}]\!]\rho hC = [\![\mathbf{P}]\!]\rho h\{\mathbf{now} - exp1 - d \# exp2 \mid \forall lineq \in C$, and $lineq = \mathbf{now} - exp1 \# exp2$, where $\# \in \{<, \leq\}, exp1, exp2$ are algebraic expressions.$\}$.

The translation of a process is a set of Horn clauses with constraints that enable us to prove that it sends certain messages or executes certain events in the constraints. The list C keeps conditions, and when they are satisfiable, the rule may fire. The translation of those events without modifying constraints is the same as [4]. So we will present the last four events in this paper.

The translation of a **Check** adds two constraints, such that the process will believe the time stamp t is new when t is in a time interval $(now - d, now)$. At that point, now in C is the time referring the clock of a pre-action before this action $Check(t)$, so it should be replaced by a new time variable old_now. P can be executed after the message has been checked and these constraints are satisfiable. The translation of a **Mark** adds a constraint, meaning that the process fetches the current time as its time stamp. The translation of a **Set** adds a mapping $d \mapsto int$ to the environment ρ. This event is used to assign the lifetime of messages, so the receiver will believe the message is new when the receiving time minus the lifetime is less than or equal to the time stamp. The translation of a **Fly** modifies constraints to loosen the lower bound made by now.

3.3 Verification

Suppose $R = \mathcal{H} \rightarrow F : C$, let GetRule be $GetRule(R) = \mathcal{H} \rightarrow F$, let GetCons be $GetCons(R) = C$, and let σ be a unifier. σ' is the maximal sub-constraint of σ involving only time terms. If L is a constraint or a set of constraints, $\sigma|_L$ is the

maximal sub-constraint of σ involving only the variables in L.

By introducing time constraints into Horn logic model, this method can verified time sensitive security protocols. The factor of time leads to rebuild processes of verification of security properties. Confidentiality requires that session keys should not be leaked by exchanging messages when they have not expired. For the sake of simplicity, we focus our attention on the verification of authentication. Our strategies are described as follows:
(1) Verifying a protocol without constraints in the Horn logic model;
(2)If there is a counter-example, we will verify the sequence of the counter-example in the extended model with time constraints. If the counter-example holds on in the extended model, then there is an attack in this protocol, else the attack is the falsehood.

[1,3,4] have given a complete work to verify protocol without constraints in the Horn logic model. So this paper will focus on how to deal with (2) in our strategies.

Definition 3.1 [Resolution] Let $R_1 = H_{11} \wedge H_{12} \wedge \cdots \wedge H_{1n} \rightarrow F : L_1$ and $R_2 = H_{21} \wedge H_{22} \wedge \cdots \wedge H_{2m} \rightarrow C : L_2$ be two logic rules, $F = attacker(Mes), H_{2i} = attacker(Mes')$, or $F = end(M, Mes), H_{2i} = end(M, Mes'), 1 \leq i \leq m$, such that Mes can be unified with Mes', and $\theta = mgu(Mes, Mes')$ is the most general unifier of Mes and Mes'. Let $L = (L_1 \cup L_2)\theta' \cup \{t_1\theta' \leq t_2\theta' | t_1 = \max\{t | t \in \mathrm{dom}(\theta'|_{L_1})\}$,and $t_2 = \max\{t | t \in \mathrm{dom}(\theta'|_{L_2})\}\}$, and if L is satisfiable, then the resolution $R_1 \bullet R_2$ between R_1 and R_2 is $(H_{21} \wedge \cdots \wedge H_{2(i-1)} \wedge (H_{11} \wedge \cdots \wedge H_{1n}) \wedge H_{2(i+1)} \wedge \cdots \wedge H_{2m})\theta \rightarrow C_2\theta : L$, we say $F' = selectedAtom(R_2)$ is the selected atom of R_2, and $\theta = sub(R_1, R_2)$ is called the substitution of the resolution $R_1 \bullet R_2$.

We assume that R_1 and R_2 don't have any other variables in common(this is simply a matter of renaming variables). R_1 provides the head, that is, it sends a message. R_2 provides a fact in the body, which means it receives a message. So the latest time in R_1 is earlier than that in R_2. Therefore, we should add a new constraint to the final constraints to represent this relation.

Definition 3.2 [Rule Implication] Let $R_1 = H_{11} \wedge \cdots \wedge H_{1m} \rightarrow C_1 : L_1$, and $R_2 = H_{21} \wedge \cdots \wedge H_{2n} \rightarrow C_2 : L_2$ be two logic rules, if $C_1 = attacker(Mes_1), C_2 = attacker(Mes_2)$, or $C_1 = end(M, Mes_1), C_2 = end(M, Mes_2)$, define rule implication $R_1 \Rightarrow R_2$, if and only if : there exists a substitution σ , such that $Mes_1\sigma = Mes_2$, and for each $H_{1i} = attacker(M_i')$, there exists $H_{2j} = attacker(M_j'')$, and for each $H_{1i} = begin(M, M_i')$, there exists $H_{2j} = begin(M, M_j'')$, such that $M_i'\sigma = M_j''(1 \leq i \leq m, 1 \leq j \leq n)$, and $L_2 \models \sigma' \wedge L_1$. The substitution σ is called the implication substitution of $R_1 \Rightarrow R_2$.

Rule implication determines whether there is an instantial relation between two rules, that is, R_2 is an instantiation of R_1.

Definition 3.3 [Derivability] Let B be a set of logic rules, let F be a closed atom, F is derivable from B if and only if there exists a finite tree defined as follows:

(1) There is one constraint C in this tree

(2) Its nodes(except the root node) are labeled by rules $R \in B$, and its edges are labeled by closed atoms.

(3) If the tree contains a node labeled by R with an incoming edge labeled by F_0 and n outgoing edges labeled by $F_1, \cdots, F_n$, then $R \Rightarrow F_1 \wedge \cdots \wedge F_n \rightarrow F_0 : C$.

(4) The root node has only one outgoing edge labeled by F.

such a tree is called a **derivation tree** of F from B. If the edges of the derivation tree are labeled by atoms instead of the closed atoms, then we say F is **weak-derivable** from B, and the tree is called a **weak-derivation tree** of F from B.

The function $GetMes$ is defined as follows: if $H = attacker(M)$, then $GetMes(H) =<$ "$attacker$", $M >$; if $H = begin(M', M)$, then $GetMes(H) =<$ "$begin$", $M >$; if $H = end(M', M)$, then $GetMes(H) =<$ "end", $M >$.

Suppose $R(= \mathcal{H} \rightarrow C : L)$, if $\mathcal{H} \neq \emptyset$, then there is a partition $\mathcal{E} = \{C_i | i = 1, \cdots, m\}(1 \leq m \leq n)$ on $\mathcal{H} = \{H_1, \cdots, H_n\}$ such that: (1)if $H_i, H_j \in C_k$, $GetMes(H_i) = GetMes(H_j)$; (2)if $i \neq j, C_i \cap C_j = \emptyset$; (3)$\forall i(1 \leq i \leq m), C_i \neq \emptyset$; (4)$\bigcup \mathcal{E} = \mathcal{H}$.

Let $\mathcal{H}' = \{H_j | j = min\{k | H_k \in C_i\}, i = 1, \cdots, m\}$, then the function $elimdup(R)$ is defined as follows: if $\mathcal{H} \neq \emptyset$, then $elimdup(R) = \mathcal{H}' \rightarrow C : L$; otherwise, $elimdup(R) = R$. If the function $elimdup$ is applied on $\mathcal{H}$, the result is $\mathcal{H}'$.

Definition 3.4 [The Sequence Of Derivation] Let P be the security protocol model on Horn logic, and R^0 be a closed rule, $R_1, \cdots, R_n \in fixpoint(P)$, if there exists $R^1, \cdots, R^n$ such that: $R^{i+1} = elimdup(R_{i+1} \bullet R^i), 0 \leq i < n$, we say $R^0, \cdots, R^n$ is a sequence of derivation from R^0 on $R_1, \cdots, R_n$, shorthand of a finite sequence of derivation in fixpoint(P).

Definition 3.5 [Goal] Atoms in the form $attacker(x)$(x is an arbitrary variable) and $begin(M, M')$ are called false goals. Atoms in the form $attacker(M')$(M' is not a variable) and $end(M, M')$ are called goals.

Definition 3.6 [Solved Form] Let $\mathcal{H} \rightarrow L : C$ be a logic rule, if the atoms in $\mathcal{H}$ are all false goals, then we say $\mathcal{H} \rightarrow L : C$ is in solved form.

Let **SolvedForm** denote the set of the logic rules that are in solved form, and **UnSolvedForm** denote the set of the logic rules that are not in solved form.

Definition 3.7 [X-resolution] Let $R_1 = H_{11} \wedge H_{12} \wedge \cdots \wedge H_{1n} \rightarrow F : L_1$ and $R_2 = H_{21} \wedge H_{22} \wedge \cdots \wedge H_{2m} \rightarrow C : L_2$ be two logic rules, $R_1 \in$ **SolvedForm**, $R_2 \in$ **UnSolvedForm**, and $F = attacker(Mes)$, let $F' = attacker(Mes')$ be a goal H_{2i} in the body of R_2, such that Mes can be unified with Mes', and $\theta = mgu(Mes, Mes')$ is the most general unifier of Mes and Mes'. Let $L = (L_1 \cup L_2)\theta' \cup \{t_1\theta' \leq t_2\theta' | t_1 = \max\{t | t \in \mathrm{dom}(\theta'|_{L_1})\}$,and $t_2 = \max\{t | t \in \mathrm{dom}(\theta'|_{L_2})\}\}$, and if L is satisfiable, then X-Resolution $R_1 \circ R_2$ between R_1 and R_2 is $(H_{21} \wedge \cdots \wedge H_{2(i-1)} \wedge (H_{11} \wedge \cdots \wedge H_{1n}) \wedge H_{2(i+1)} \wedge \cdots \wedge H_{2m})\theta \rightarrow C\theta : L, F' = selectedGoal(R_2)$ is called the selected goal of $R_2, \theta = sub(R_1, R_2)$ is called the substitution of the X-Resolution $R_1 \circ R_2$.

Let $R_0 = R_2 \circ R_3, R_1 = elimdup(R_0), R_2 \in$ **UnSolvedForm**, $R_3 \in$**SolvedForm**, $\theta = sub(R_2, R_3)$, for all $i \in \{0,1,2,3\}, R_i = H_{i1} \wedge \cdots \wedge H_{if(i)} \rightarrow C_i : L_i$, where $f(i)$ denotes the number of atoms in the body of $R_i, H_{ij} = attacker(U_{ij})$ or $H_{ij} = begin(M_{ij}, U_{ij})(1 \leq j \leq f(i))$, for each $U_{1j}(1 \leq j \leq f(1))$, define representativeAtom(U_{1j}) and deletedAtom(U_{1j}) as follows:
representativeAtom(U_{1j}) $= H_{0s}$, where $s = min\{k | U_{0k}\theta = U_{1j}, k = 1, \cdots, f(0)\}$;
deletedAtom(U_{1j}) $= \{H_{0k} | U_{0k}\theta = U_{1j}\}-$ representativeAtom(U_{1j}).

Let $R_1 = \mathcal{H}_1 \rightarrow F_1 : L_1$ and $R_2 = \mathcal{H}_2 \rightarrow F_2 : L_2$ be two logic rules. In the definition of resolution, each atom such as $attacker(M')$ and $end(M, M')$ in the body of R_2 can be selected as a selected atom; in the definition of X-resolution, false goals will not be selected. Let R be a logic rule, and B be a set of logic rules. We can define addRule(R,B) as follows:

If $\exists R' \in B, R' \Rightarrow R$, then addRule(R,B)=B;

else addRule(R,B)=$\{R\} \cup \{R' | R' \in B, R \not\Rightarrow R'\} \cup \{marked(R'') | R'' \in B, R \Rightarrow R''\}$.

Let $\{R_1, \cdots, R_n\}$ be a set of logic rules, define

$$addRule(\{R_1, \cdots, R_n\}, B) = addRule(\{R_2, \cdots, R_n\}, addRule(R_1, B)).$$

Marked(R'') denotes that R'' will not be used to compute X-Resolutions. Let Marked denote the set of logic rules that will not be used to compute X-Resolutions, and UnMarked denote the set of logic rules that are not in Marked.

Let P be a logic program, define:
$Rule^{(0)}(P) = \{elimdup(R) | R \in P\}$
$T^{(0)}(P) = Rule^{(0)}(P) \cap$ **SolvedForm**
$C^{(0)}(P) = Rule^{(0)}(P) \cap$ **UnSolvedForm**
$X\_Resolution^{(0)}(P) = \{elimdup(R) | R = R' \circ R'', R' \in T^{(0)}(P), R'' \in C^{(0)}(P)\}$
$Rule^{(n+1)}(P) = addRule(X\_Resolution^{(n)}(P), rule^{(n)}(P))$
$T^{(n+1)}(P) = Rule^{(n+1)}(P) \cap$ **SolvedForm**
$C^{(n+1)}(P) = Rule^{(n+1)}(P) \cap$ **UnSolvedForm**
$X\_Resolution^{(n+1)}(P) = \{elimdup(R) | R = R' \circ R'', R' \in T^{(n)}(P), R'' \in C^{(n)}(P)\}$.

Definition 3.8 [Solved-Form fixpoint] Let P be the model of a security protocol on Horn logic, define $fixpoint(P) = \{T^{(n)}(P) | n \geq 0\} \cap$**UnMarked**, $fixpoint(P)$ is called the Solved-Form fixpoint of P.

For the authentication property, define derivablerec(R,B,P) as:

if $\exists R' \in B, R' \Rightarrow R$, then $derivablerec(R, B, P) = \emptyset$

else if $R = begin(M_1, M_1') \wedge \cdots \wedge begin(M_n, M_n') \rightarrow end(M, M')$, then

$$derivablerec(R, B, P) = \{R\}$$

else $derivablerec(R, B, P) = \bigcup \{derivablerec(elimdup(R' \bullet R), \{R\} \cup B, P) | R' \in fixpoint(P)\}$

And define $derivable(F, P) = derivablerec(F \rightarrow F : true, \emptyset, P)$.

Lemma 3.9 *Let P be the model of a security protocol on Horn logic and F be a closed atom. If F is weak-derivable from P, then F is derivable from P.*

Let C and S be two constraints systems, and Q be a set of variables, define: $find\_corr\_exp(C, Q) = \{exp | exp \in C,$ and $fv(exp) \cap Q \neq \emptyset\}$ and $correspond\_exp(C, S) = \{exp | exp \in C,$ and $fv(exp) \cap fv(S) \neq \emptyset\}$ Obviously, there exists $n > 0$ such that $\sqcup(correspond\_exp_C(C))(S) = (correspond\_exp_C(C))^n(S)$, where $correspond\_exp_C(C)(S)$ is the curried function of $correspond\_exp(C, S)$.

Definition 3.10 [Derivation sub-tree] Let P be the extended Horn logic model of a security protocol and F be a closed atom, T is a derivation tree of F whose constraints tag is C. T' is called a derivation sub-tree of T if T', whose constraints tag is $\sqcup\big(correspond\_exp_C(C)\big)\big(find\_corr\_exp(C,\ fv(T'))\big)$, is obtained by cutting some edges in T and all branches starting from these edges. If T' has n nodes, then T' is called an n-nodes derivation sub-tree of T.

Lemma 3.11 *Let P be the model of a security protocol on Horn logic, and F be a closed atom, where F is derivable from P and T is a derivation tree of F. Let T' be a derivation sub-tree of T and C be the constraint of T',T' is obtained by cutting edges $e_1, \cdots, e_s$ in T and all branches starting from $e_1, \cdots, e_s$, which are labeled by closed atoms $F_1, \cdots, F_s$ respectively, then there exists a logic rule R such that $R \Rightarrow F_1 \wedge \cdots \wedge F_s \rightarrow F : C$, moreover, there exists a logic rule R' and a logic rule set B such that $derivablerec(R', B, P) \subseteq derivablerec(F, P)$ and $\forall R'' \in B, R'' \not\Rightarrow R'$.*

The authentication properties are characterized by the correspondence assertions $begin(M, M')$ and $end(M, M')$, let $B_b = \{\rightarrow begin(M_1, M_1'), \cdots, \rightarrow begin(M_n, M_n')\}$, and for each $begin(M, M')$ or $end(M, M')$, there exists a time variable $t_{begin(M,M')}$ or $t_{end(M,M')}$.

Definition 3.12 [Authentication] Let P be the security protocol's model on Horn logic, $begin(M, M')$ and $end(M, M')$ be the correspondence assertions, and $end(M, M')$ be a closed atom, if $end(M, M')$ is derivable from $P \cup B_b$, then $\rightarrow begin(M, M') \in B_b$, and $t_{end(M,M')} - t_{begin(M,M')} < \Delta$($\Delta$ is the allowable delay in the protocol), we say the security protocol satisfies the authentication property with respect to $begin(M, M')$ and $end(M, M')$.

Theorem 3.13 *Let P be the model of a security protocol on Horn logic, and let F be a closed atom as $end(M, M')$, if F is derivable from $fixpoint(P) \cup B_b$, then $begin(M, M') \in B_b$, if and only if, if F is derivable from $P \cup B_b$, then $begin(M, M') \in B_b$.*

The proofs of those lemmas and theorem are similar as those in [3], and we just modify the proofs trivially to get proofs of these theorems. Because some definitions are added with time constraints, we also require these constraints be satisfiable as described by definitions when we prove them. The strategies of proofs are the same as those in [3]. These modifications are very easy, so we will not give out these proofs.

Theorem 3.14 *Let P be the model of a security protocol on Horn logic, and let F be a closed atom as $end(M, M')$, F is derivable from $fixpoint(P) \cup B_b$, and $begin(M, M') \in B_b$, if and only if, there exists $H_1 \wedge \cdots \wedge H_n \rightarrow F : C \in derivable(F, P)$, where each $H_i \in B_b (1 \leq i \leq n)$ is the atom as $begin(M_i, M_i')$, $begin(M, M') \in \{H_1, \cdots, H_n\}$ and C is solvable.*

4 Constraints system

From the analysis of our method, there is no problem in verifying time sensitive security protocols in Horn logic model. But it is also important to determine whether the constraints system has a solution or not. Though there are so many methods to solve linear constraints, such as the method of Simplex and the algorithm of Fourier-Motzkin [8][22], these methods pay attention to the general linear systems and are complicated to be implemented. When there are symbolic arguments in constraints system, it is very difficult to determine the solution by these methods. Therefore, this section will discuss the satisfiability of our constraints system.

Definition 4.1 A TVPI system is a system of linear inequalities where each inequality involves at most two variables. A TVPI system is called **monotone** if each inequality is of the form $ax_i - bx_j \leq c$, where both a and b are positive. If $a = b = 1$, we call this system is a uni-TVPI system.

In our system, inequalities can be generated by the following case:

- **Check** event: this event will check if the time stamp t is valid, so $\mathbf{now} - d < t < \mathbf{now}$ will be added to constraints system C. This event introduces inequalities that are of the form $x - y < c$ which can be loosed into the form of $x - y \leq c$.
- **Mark** event: this event only generate equality $t = \mathbf{now}$.
- **Set,Fly** events: they only modify time constants, and don't add or minus inequalities in our system.
- The process of resolution and X-resolution will introduce a new inequality $t_1\theta' \leq t_2\theta'$ to the merge of two uni-TVPI systems, so the final system is a uni-TVPI one.

Therefore, our constraints system is a uni-TVPI system. we can use the Fourier-Motzkin elimination method to find a feasible solution to a uni-TVPI system. However, the main drawback of this method is that the number of inequalities may grow exponentially. In a uni-TVPI system, we can make a stronger statement:

Theorem 4.2 *A uni-TVPI system $\mathcal{S}$ has no solution if and only if there is a sequence $t_1 \geq t_2 + d_1, t_2 \geq t_3 + d_2, \cdots, t_n \geq t_1 + d_n$, and $\sum_{i=1}^{n} d_i > 0$.*

To determine whether or not a uni-TVPI system has solution, our strategy is as follows:

Suppose t_i has a greatest lower bound $t_{i+1} + d_i (i = 1, \cdots, n)$, and $t_{n+1} = t_1$. Using resolution, one obtains a sequence of inequalities $t_1 \geq t_{i+1} + \sum_{j=1}^{i} d_j (i = 1, \cdots, n)$.

The last one will yield $0 \geq \sum_{i=1}^{n} d_i$. *If* $\sum_{i=1}^{n} d_i$ *is actually greater than 0, there is a contradiction, that is, the system can't be satisfied.*

Theorem 4.3 *A uni-TVPI system* $\mathcal{S}$ *has no solution if and only if there is a cycle* $t_1 \xrightarrow{d_1} t_2 \xrightarrow{d_2} \cdots \xrightarrow{d_{n-1}} t_n \xrightarrow{d_n} t_1$ *in the corresponding graph* G, *and* $\sum_{i=1}^{n} d_i > 0$.

This problem can be translated into one in the graph, and it is very easy to implement the algorithm for the solution of the constraints system. We present a polynomial algorithm to find a cycle $t_1 \xrightarrow{d_1} t_2 \xrightarrow{d_2} \cdots \xrightarrow{d_{n-1}} t_n \xrightarrow{d_n} t_1$ in the corresponding graph G, and $\sum_{i=1}^{n} d_i > 0$ in [18].

5 Detailed Analysis of Protocol

In this section, we will show how to verify time sensitive security protocols in our method. For convenience, we use $<$ instead of $\leq$ in our model. Based on the translation in Section 3.2, the logic program of Denning-Sacco protocol can be derived as follows:

A:① $\rightarrow attacker(A) : true$

② $\rightarrow attacker(B) : true$

③ $begin(\cdots) \wedge attacker(\{B, XKab, XT1, XTicket\}Kas) \rightarrow attacker(XTicket) : now - d1 < XT1 < now$

④ $begin(\cdots) \wedge attacker(\{B, XKab, XT1, XTicket\}Kas) \wedge attacker(\{N_B(t)\}XKab) \rightarrow attacker(\{N_B(t) - 1\}XKab) : now - d1 - (d1 + d2) < XT1 < now$

S:⑤ $attacker(x) \wedge attacker(y) \rightarrow attacker(\{y, Kab(T1), T1, \{Kab(T1), T1, x\}Kys\}Kxs : T1 = now$

B:⑥ $attacker(\{XKab, XT1, xA\}Kbs) \rightarrow attacker(\{N_B(XT1)\}XKab) : now - d1 - d2 < XT1 < now$

⑦ $attacker(\{XKab, XT1, xA\}Kbs) \wedge attacker(\{N_B(XT1) - 1\}XKab) \rightarrow end(\cdots) : now - 2(d1 + d2) < XT1 < now$

I:⑧ $\rightarrow attacker(Kab(t)) : now > D + t$

where D is the lifetime of session key Kab and is a very big constant, we can suppose D is bigger than all of time constraints. So we will resolve as follows:

①②⑤:(1) $\rightarrow attacker(\{B, Kab(T1), T1, \{Kab(T1), T1, A\}Kbs\}Kas) : T1 = now$

③(1):(2) $begin(\cdots) \rightarrow attacker(\{Kab(t1), t1, A\}Kbs) : now - d1 < t1 < now$

⑥(2):(3) $begin(\cdots) \rightarrow attacker(\{N_B(t1)\}Kab(t1)) : now - d1 - d2 < t1 < now,$
$t2 - d1 < t1 < t2 < now$

(4) $attacker(Key) \wedge attacker(\{X\}Key) \rightarrow attacker(X) : true$

⑧(4):(5) $attacker(\{X\}Kab(t)) \rightarrow attacker(X) : now > D + t$

(3)(5):(6) $begin(\cdots) \rightarrow attacker(N_B(t1)) : now > D + t1, t3 - d1 - d2 < t1 < t3,$
$t2 - d1 < t1 < t2 < t3 < now$

(7) $attacker(x) \rightarrow attacker(f(x)) : true$, where $f : x \mapsto x - 1$

(6)(7):(8) $begin(\cdots) \rightarrow attacker(f(N_B(t1))) : t4 > D + t1, t3 - d1 - d2 < t1 < t3,$
$t2 - d1 < t1 < t2 < t3 < t4$

(9) $attacker(x) \wedge attacker(K) \rightarrow attacker(\{x\}K) : true$

⑧(9):(10) $attacker(x) \rightarrow attacker(\{x\}Kab(t)) : t5 > D + t$

(8)(10):(11) $begin(\cdots) \rightarrow attacker(\{f(N_B(t1))\}Kab(t)) : t4 > D + t1, t3 - d1 - d2 <$
$t1 < t3, t2 - d1 < t1 < t2 < t3 < t4, t5 > D + t, t5 > t4$

⑦(2):(12) $begin(\cdots) \wedge attacker(\{N_B(t1) - 1\}Kab(t1)) \rightarrow end(\cdots) : t2 - d1 < t1 <$
$t2, now - 2(d1 + d2) < t1 < now$

(11)(12):(13) $begin(\cdots) \rightarrow end(\cdots) : t2' - d1 < t1 < t2', now - 2(d1 + d2) < t1 <$
$now, t4 > D + t1, t3 - d1 - d2 < t1 < t3, t2 - d1 < t1 < t2 < t3 < t4, t5 >$
$D + t1, t5 > t4$

By Section 4, we have $t1 > now - 2(d1 + d2) > t5 - 2(d1 + d2) > D + t1 - 2(d1 + d2) = t1 + (D - 2(d1 - d2))$, that is, $2(d1 + d2) > D$. If this symbolic condition is satisfied, there is no contradiction. In fact, the lifetime of a message is far smaller than the lifetime of the session key, so there is $2(d1 + d2) < D$. It means that the constraints system of (13) has no solution. The resolution couldn't continue, so the freshness attack in Needham-Schroeder protocol do not appear in Denning-Sacco protocol. The meaning of above steps is that the rule (2) means A sends a new ticket which B receives. When B believes this ticket is a new one, he gives a freshness nonce encrypted by the session key. The rule ⑦ limits time stamp $XT1$(which is $t1$ later) in a special interval, that is $now - 2(d1 + d2) < t1 < now$ in the (13). When we gain the rule (12), the constraints assume the current clock is greater than $D + t1$. The constraints of the last rule require the distance between t1 and the current time is less than $2(d1 + d2)$, and also the current time is greater than $D + t1$, but $D \geq 2(d1 + d2)$, so these constraints will not be satisfied.

6 Conclusions

The verification on Horn logic is very effective, and it can verify the interleaving run of security protocol's infinite sessions. But this model cannot verify time sensitive security protocols, because there is no time information in them. This paper takes as examples the linear arithmetic constraints in [9], and add time constraints to the Horn model. [9] verifies protocols in backward exploration, so it defines a special

term **sup** which is not easy to deal with in unification. The resolution of Horn logic is faster than those of other logics. The complexity in the Horn logic model with time constraints just increases in the deciding the satisfiability of constraints. With time constraints, we provide a uniform framework for the specification of an arbitrary number of protocol sessions with fresh name generation and time stamps. Denning-Sacco protocol is verified in this paper. The result illustrates that, firstly, our model with time constraints reserves the efficiency of the Horn logic model, and secondly, that it extends the capability of analysis and verification. In this method, we can find the condition in which there is no attack in the protocol. People found some protocols' flaws which may not satisfy time constraints, that is, these flaws are false attacks. Therefore, analyzing and verifying time sensitive security protocols is an effective way to avoid false attacks. Our approach is an innovative and effective method on verifying time sensitive security protocols.

Our future work is : (1) to implement this method in our verifying tool SPVT to verify time sensitive security protocols and to give a condition in which there is a solution; (2) to use the improved SPVT to automatically verify sophisticate time sensitive protocols such as Kerberos V.

Acknowledgement

We would like to thank Prof. Xiaoyu Song for useful discussions about the structure of this paper and useful modifications for its presentation. We would also like to thank the anonymous referees, whose useful comments have greatly improved the quality of this paper.

References

[1] Abadi, M. and B. Blanchet, *Analyzing security protocols with secrecy types and logic programs*, in: *Symposium on Principles of Programming Languages*, 2002, pp. 33–44.
URL citeseer.ist.psu.edu/abadi02analyzing.html

[2] Bella, G., "Inductive Verification of Cryptographic Protocols," Ph.d. thesis, University of Cambridge, London (2000).
URL citeseer.ist.psu.edu/article/bella00inductive.html

[3] Blanchet, B., *An efficient cryptographic protocol verifier based on prolog rules*, in: *14th IEEE Computer Security Foundations Workshop (CSFW-14)*, 2001, pp. 82–96.

[4] Blanchet, B., *From secrecy to authenticity in security protocols*, in: M. Hermenegildo and G. Puebla, editors, *9th International Static Analysis Symposium (SAS'02)*, Lecture Notes in Computer Science **2477** (2002), pp. 342–359.

[5] Bozga, L., C. Ene and Y. Lakhnech, *A symbolic decision procedure for cryptographic protocols with time stamps (extended abstract).*, in: *CONCUR*, 2004, pp. 177–192.

[6] Clark, J. A. and J. L. Jacob, *A survey of authentication protocol literature*, Technical Report 1.0 (1997).
URL citeseer.ist.psu.edu/clark97survey.html

[7] Corin, R., S. Etalle, P. H. Hartel and A. Mader, *Timed model checking of security protocols*, in: *FMSE '04: Proceedings of the 2004 ACM workshop on Formal methods in security engineering* (2004), pp. 23–32.

[8] Cormen, T. H., C. E. Leiserson, R. L. Rivest and C. Stein, "Introduction to Algorithms," MIT Press, Cambridge, Massachusetts, 2001, second edition.

[9] Delzanno, G. and P. Ganty, *Automatic Verification of Time Sensitive Cryptographic Protocols*, in: *Proc. of Int. Conf. on Tools and Algorithms for Construction and Analysis of Systems (TACAS'04)*, Lecture Notes in Computer Science **2988** (2004), pp. 342–356.

[10] Denning D, S. G., *Timestamps in key distribution protocols*, Communications of the ACM **24** (1981), pp. 533–536.

[11] Dolev, D. and A. C. Yao, *On the security of public key protocols*, Technical report, Stanford, CA, USA (1981).

[12] Evans, N. and S. Schneider, *Analysing time dependent security properties in CSP using PVS*, in: *ESORICS*, 2000, pp. 222–237.

[13] Gorrieri, R., E. Locatelli and F. Martinelli, *A simple language for real-time cryptographic protocol analysis.*, in: *ESOP*, 2003, pp. 114–128.

[14] Hui, M. L. and G. Lowe, *Fault-perserving simplifying transformations for security protocols*, Journal of Computer Security **9** (2001), pp. 3–46.

[15] Lee, J. Y. and J. Zic, *On modeling real-time mobile processes*, in: *ACSC '02: Proceedings of the twenty-fifth Australasian conference on Computer science* (2002), pp. 139–147.

[16] Li, M., Z. Li and H. Chen, *Security protocol's extended horn logic model and its verification method*, Chinese Journal of Computers **29** (2006), pp. 1667–1678.

[17] Li, M., Z. Li and H. Chen, *Spvt: An efficient verification tool for security protocol*, Chinese Journal of Software **17** (2006), pp. 898–906.

[18] Li, Z., T. Zhou, M. Li and H. Chen, *Constraints solution for time sensitive security protocols*, in: *FAW 2007*, Lecture Notes in Computer Science **4613** (2007), pp. 190–202.

[19] Lowe, G., *Casper: A compiler for the analysis of security protocols*, in: *10th IEEE Computer Security Foundations Workshop (CSFW-10)* (1997), pp. 18–30.

[20] Lowe, G. and J. Ouaknine, *On timed models and full abstraction.*, Electr. Notes Theor. Comput. Sci. **155** (2006), pp. 497–519.

[21] Needham R, S. M., *Using encryption for authentication in large networks of computers*, Communications of the ACM **21** (1978), pp. 993–999.

[22] Schrijver, A., "Theory of Linear and Integer Programming," John Wiley and Sons, 1986.

[23] Sewell, P., *Applied π – a brief tutorial*, Technical Report UCAM-CL-TR-498, Computer Laboratory, University of Cambridge (2000), 65pp.
URL http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-498.html

[24] Syverson, P. F., *Adding time to a logic of authentication*, in: *CCS '93: Proceedings of the 1st ACM conference on Computer and communications security* (1993), pp. 97–101.

[25] Zhou, T., M. Li, Z. Li and H. Chen, *Automatically constructing counter-examples of security protocols based on the extended Horn logic model*, Chinese Journal of Computer Research and Development **44** (2007), pp. 1518–1531.