

# Confidentiality-preserving Proof Theories for Distributed Proof Systems

Kazuhiro Minami  
National Institute of  
Informatics  
minami@nii.ac.jp

Nikita Borisov  
University of Illinois at  
Urbana-Champaign  
nikita@illinois.edu

Marianne Winslett  
University of Illinois at  
Urbana-Champaign  
winslett@illinois.edu

Adam J. Lee  
University of Pittsburgh  
adamlee@cs.pitt.edu

## ABSTRACT

A distributed proof system is an effective way for deriving useful information by combining data from knowledge bases managed by multiple different principals across different administrative domains. As such, many researchers have proposed using these types of systems as a foundation for distributed authorization and trust management in decentralized systems. However, to account for the potentially sensitive nature of the underlying information, it is important that such proof systems be able to protect the confidentiality of the logical facts and statements.

In this paper, we explore the design space of sound and safe confidentiality-preserving distributed proof systems. Specifically, we develop a framework to analyze the theoretical best-case proving power of these types of systems by analyzing confidentiality-preserving proof theories for Datalog-like languages within the context of a trusted third party evaluation model. We then develop a notion of safety based on the concept of non-deducibility and analyze the safety of several confidentiality-enforcing proof theories from the literature. The results in this paper show that the types of discretionary access control enforced by most systems on a principal-to-principal basis are indeed safe, but lack proving power when compared to other systems. Specifically, we show that a version of the Minami-Kotz (MK) proof system can prove more facts than the simple DAC system while retaining the safety property of the simple system. We further show that a seemingly-useful modification of the MK to support commutative encryption breaks the safety of the system without violating soundness.

**Categories and Subject Descriptors:** C.2.4 [Distributed Systems]: Distributed applications; K.6.5 [Management of Computing and Information Systems]: Security and Protection

**General Terms:** Security

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ASIACCS '11, March 22–24, 2011, Hong Kong, China.

Copyright 2011 ACM 978-1-4503-0564-8/11/03 ...\$10.00.

**Keywords:** Access control, inference control, distributed proving, Datalog

## 1. INTRODUCTION

Computer systems that span multiple administrative domains are increasingly interdependent, both in the context of virtual business coalitions and in pervasive computing systems. A distributed proof system is an effective way to combine information held by different principals spanning these domains. Many researchers have proposed using these techniques as a foundation for distributed authorization and trust management systems [1, 2, 4, 6, 8, 9, 11, 13].

Such distributed proof systems can combine information across disparate administrative domains, and thus must provide a means of controlling the flow of sensitive information between these domains. However, as some underlying data may be sensitive, it is important to enforce confidentiality policies to control control which principals are allowed to make use of confidential information stored within any principal's knowledge base. For example, in virtual business coalitions, organizational attributes are frequently sensitive and should be revealed only to trusted parties.

Despite the obvious importance of confidentiality, it is not modeled *explicitly* by most distributed proof construction systems. Rather, there is an *implicit* assumption that all principals will enforce their confidentiality policies by controlling what is directly revealed in two-party interactions among principals. In particular, inferences that may be made based on the external state of a system are not explicitly modeled. Consider an example proof system with three principals  $p_0$ ,  $p_1$ , and  $p_2$  with the following knowledge bases respectively.<sup>1</sup>

$$KB_0 = \{f_0\},$$

$$KB_1 = \{f_1 \leftarrow p_0 \text{ says } f_0\}$$

$$KB_2 = \{f_2 \leftarrow p_1 \text{ says } f_1\}.$$

Suppose principal  $p_0$  is willing to reveal fact  $f_0$  to  $p_1$  but *not*  $p_2$ . With pairwise policy enforcement,  $p_2$  will be able to derive  $f_2$  after  $p_1$  derives  $f_1$  and reveals this to  $p_2$ . Is this process safe, or might  $p_2$  learn something about the truth of  $f_0$  through inference, in contradiction to the confidentiality policy? Conversely, suppose that  $p_0$  is willing to reveal  $f_0$

<sup>1</sup>We assume a Datalog-like policy language here, and describe the details of this language in Section 2.

to  $p_2$  but *not*  $p_1$ . Pairwise policy enforcement will not allow any derivations to go through; however, a distributed cryptographic protocol developed by Minami and Kotz [14] will allow  $p_2$  to derive  $f_2$ , while keeping  $p_1$  in the dark, by sending part of the proof in an encrypted form. Even though no obvious violations of confidentiality policies, how can we know whether this approach is safe?

To answer these questions, we develop a new definition of confidentiality that models the *global* properties of the system—rather than just local interactions—and captures not only direct revelation of knowledge between principals, but also *inferences* that can be drawn. Our definition is based on the concept of nondeducibility articulated by Sutherland in [15]. Informally, this states that a principal is unable to infer the value of a confidential fact when, given her partial view of the system, there exists one possible valid configuration of the remainder of the system in which the fact in question is true, and another in which the fact is false. We extend this simple definition to capture inferences that can be drawn across the relationships between the values of several confidential facts, and to take into account potential collusion among malicious parties.

In our analysis, we model a distributed proof system using a proof theory consisting of a set of inference rules for deriving new information. This allows us to abstract away protocol details and focus instead on what is *provable* in a distributed system. Analysis based on a proof theory gives us insights into the “ideal” functionality of the proof system, and allows us to establish a theoretical upper bound for what can be derived using any safe distributed proof system. In exploring this problem, we make the following contributions:

1. We develop formal definitions of confidentiality and safety for distributed proof systems based on the notion of *nondeducibility*.
2. We present a safety analysis demonstrating the existence of a safe distributed proof system that derives more logical statements than a system that simply enforces confidentiality policies locally.
3. We further describe an unsafe result for a proof system that supports commutative encryption, which shows the effectiveness of our theoretical framework for safety analysis.

The rest of the paper is organized as follows. We introduce a reference model for distributed proof systems parameterized by a set of inference rules in Section 2. We introduce our attack model and develop a formal definition of safety that considers inference attacks by malicious colluding principals in Section 3. We analyze several different confidentiality-preserving proof systems in Section 4. We discuss related work in Section 5 and conclude in Section 6.

## 2. SYSTEM MODEL AND DEFINITIONS

In this section, we first describe the system model in which we will study distributed proof construction. We then show how systems for enforcing disclosure policies placed on sensitive facts can be represented as sets of inference rules augmenting the base logical framework. Finally, we introduce a computational model based on a trusted third party, which provides us with a theoretical framework for defining the formal notion of safety used throughout the rest of the paper.

## 2.1 System Environment

Many existing distributed proof systems use Datalog to define their semantics (e.g., Delegation logic [12], DKAL [8], SD3 [9], SecPAL, and Gray [2]). As such, we treat Datalog as our underlying logical language. We assume a distributed system consisting of a set  $\mathcal{P}$  of principals, each of whom autonomously maintains a Datalog knowledge base containing sets of facts, quoted facts, and rules. A *fact* is a predicate symbol followed by zero or more terms, where each term is either a lower-case or numerical constant, or a variable (denoted by an upper-case letter). For example, the fact *location(alice, 1120)* indicates that Alice is currently located in room 1120. *Quoted facts* are used to represent knowledge derived by other principals in the system, and make use of the *says* modal operator. For example, the quoted fact (*hr says grad(bob)*) indicates that Bob is a graduate student (i.e., *grad(bob)*) according to the knowledge base maintained by the human resource department (i.e., *hr*). We denote the sets of facts and quoted facts by using the symbols  $\mathcal{F}$  and  $\mathcal{Q} = \mathcal{P} \times \mathcal{F}$ , respectively.

Datalog *rules* are clauses of the form  $f \leftarrow q_1, \dots, q_n$  in which  $f$  is a fact and each  $q_i$  is a quoted fact. These rules allow a principal to derive new facts based upon facts in her knowledge base and the knowledge bases of others. For example, consider the following rule:

$$\begin{aligned} \text{grant}(U, db) &\leftarrow \text{role}(U, \text{doctor}), \\ &\quad ls \text{ says } \text{location}(U, \text{hospital}) \end{aligned}$$

This rule states that any user  $U$  can access the database  $db$  if  $U$  is assigned to the “doctor” role and the location service (i.e.,  $ls$ ) indicates that  $U$  is located in the hospital. Since any fact  $f_i$  in the knowledge base of principal  $p_i$  can trivially be represented as the quoted fact ( $p_i \text{ says } f_i$ ), we denote the set of all Datalog rules as  $\mathcal{R} \subseteq \mathcal{F} \times 2^{\mathcal{Q}}$ . We assume that the rules contained in a principal’s knowledge base are private, and that facts are only exchanged in accordance with the rules set forth by the *specific* proof construction approach (see Section 2.2). Finally, to ensure that principals can securely exchange quoted facts, we assume that any two principals  $p_i, p_j \in \mathcal{P}$  can establish a private and authenticated communication channel between them by using a PKI or some other key distribution channel.

## 2.2 Inference Rules

We model the specifics of a distributed proof system as a collection of *inference rules* for deriving new facts in each principal’s knowledge base. Inference rules are *meta-rules* that are independent of any particular Datalog rules stored in the knowledge bases, and describe a syntactic mapping from given Datalog rules and facts into new facts to be derived. In standard Datalog, the *Elementary Production Principle (EPP)* is the only inference rule for deriving new facts [5]:

$$\text{(EPP)} \frac{f_0 \leftarrow f_1, \dots, f_n \quad f_1 \quad \dots \quad f_n}{f_0}$$

According to the (EPP) rule, if there is a rule  $f_0 \leftarrow f_1, \dots, f_n$  and every fact  $f_i$  in the body of the rule exists, then  $f_0$  will be derived.

In a distributed setting, inference rules need to explicitly account for facts being managed in many knowledge bases. In the most basic distributed Datalog system, all facts are

$$\begin{aligned}
(\text{COND}) \quad & \frac{(f \leftarrow q_1, \dots, q_n) \in KB_i \quad q_k \in KB_i \text{ for all } k}{f \in KB_i} \\
(\text{SAYS}) \quad & \frac{f \in KB_i}{(p_i \text{ says } f) \in KB_j}
\end{aligned}$$

**Figure 1: Inference rules parameterizing the reference distributed proof system  $D[I_S]$**

shared freely between principals as indicated in Figure 1. The inference rule (COND) allows each principal  $p_i$  to locally apply the Datalog (EPP) rule to derive a new fact  $f$  in its knowledge base  $KB_i$ . Note that a rule in the premise of (COND) contains quoted facts  $q_1, \dots, q_n$  in its body, thus  $p_i$  can derive a new fact from statements made by remote principals. The inference rule (SAYS) allows a principal  $p_j$  to import a quoted fact  $(p_i \text{ says } f)$  into its knowledge base  $KB_j$  if principal  $p_i$  maintains a fact  $f$  in its knowledge base  $KB_i$ . We denote this reference proof system as  $D[I_S]$ , signifying that it represents a basic Datalog system augmented by distributed inference rules  $I_S = \{(\text{COND}), (\text{SAYS})\}$ . We will use  $D[I_S]$  as a point of comparison when evaluating other confidentiality-preserving distributed proof systems later in the paper. The confidentiality-preserving distributed proof systems in Section 4 will replace the inference rule (SAYS) with a rule with additional conditions to satisfy confidentiality policies of the principals in a system.

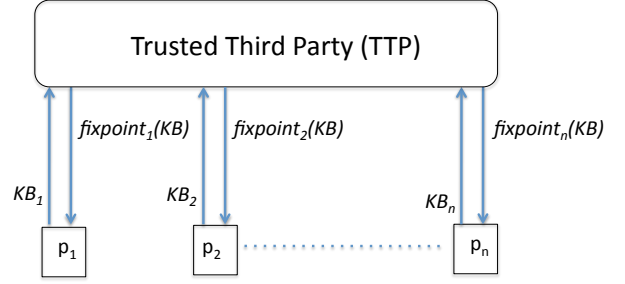
### 2.3 Computation with a Trusted Third Party

A distributed proof system  $D[I]$  updates a set of knowledge bases  $KB$  by deriving new facts or quoted facts using inference rules in set  $I$  iteratively in a bottom-up way. In this paper, we are interested primarily in what is provable at the *final* state of a distributed proof system—i.e., once all possible inference have been made—as this gives us a theoretical upper bound on what the system can prove, regardless of the communication and implementation details of the system itself. To this end, we adopt a system model that abstracts away these details as shown in Figure 2. In this model, a trusted third party (TTP) collects the set of initial knowledge bases  $KB$  from the principals, simulates the execution of the original proof system  $D[I]$  using the inference rules  $I$ , and returns to each principal  $p_i$  a knowledge base  $KB_i$  representing its final state.

To define the behavior of the system  $D$  precisely, we introduce the immediate consequence operators  $T_{I,i}$  and  $T_I$ , as follows:

$$\begin{aligned}
T_{I,i}(KB) &= \{q \mid (q, p_i) \in \text{infer}(KB, I)\} \cup KB_i \\
T_I(KB) &= (T_{I,1}(KB), \dots, T_{I,n}(KB))
\end{aligned}$$

The function  $T_{I,i}$  takes a set of knowledge bases  $KB$  as input and outputs a principal  $p_i$ 's knowledge base at the next step. This is accomplished by applying the *infer* function, which takes a set of knowledge bases  $KB$  and a set of inference rules  $I$  as inputs and outputs a set  $S$  of tuples of the form  $(q, p_i)$  if principal  $p_i$  derives a quoted fact  $q$  (i.e.,  $q \in KB_i$ ) in one step by applying an inference rule in  $I$  to rules and facts in the  $KB$ . The immediate consequence operator  $T_I$  extends this notion to the knowledge bases of all principals in the obvious way.



**Figure 2: Trusted third party (TTP) computational model for distributed proof systems.**

A distributed proof system  $D[I]$  can thus be represented as a state transition system in which states are represented by sets of knowledge bases, and transitions are entailed by the immediate consequence operator  $T_I$ . In Datalog, this state transition system is guaranteed to reach a fixpoint in which no new facts can be derived using the inference rules  $I$ , namely the least Herbrand model for the given rules and facts [5]. Therefore, our reference proof system  $D[I_S]$  will always have a fixpoint for all possible initial states of its knowledge bases  $KB$ , as quoted facts in a distributed proof system can be converted easily into ordinary Datalog facts, which take a principal identity as an additional parameter.<sup>2</sup> We are now able to define the TTP representation of a distributed proof system as follows:

**DEFINITION 1 (DISTRIBUTED PROOF SYSTEM).** *Let  $\mathcal{KB} \equiv \mathcal{F} \cup \mathcal{Q} \cup \mathcal{R}$  represent the set of all possible knowledge bases. Given a finite set  $P$  of principals, a distributed proof system  $D[I]$  based on the TTP model is defined as the function  $\text{fixpoint}[I] : \mathcal{KB}^{|P|} \rightarrow \mathcal{KB}^{|P|}$  where  $\text{fixpoint}[I](KB)$  is the transitive closure of the immediate consequence operator  $T_I$  on an initial set of knowledge bases  $KB \in \mathcal{KB}^{|P|}$ .*

In the remainder of this article, we study the safety of distributed proof systems using this TTP model. Although such a TTP could be simulated using secure multi-party computation techniques [7], this is not our intention. Rather, we use this construction to reason about the theoretical upper bounds on safety and provability that can be attained in the distributed proof construction setting. This can then be used as a target for system designers interested in building efficient, yet secure, distributed proof systems.

### 2.4 Soundness

In this paper, we consider a class of distributed proof systems that have the same model semantics as defined by the function  $\text{fixpoint}[I_S]$ .

**DEFINITION 2 ( $\models$ , MODEL).** *We say  $KB \models (q \in KB_i)$  if  $(q, p_i) \in \text{fixpoint}_i[I_S]$ .*

However, since each distributed proof system may be described by a different set of inference rules, the facts that are provable from some initial knowledge base may differ from system to system. We now formally define the notion of provability.

<sup>2</sup>For example the quoted fact  $(ls \text{ says } \text{location}(\text{alice}, 1120))$  can be represented as the fact  $\text{location}(ls, \text{alice}, 1120)$ .

DEFINITION 3 (PROOF). Consider a distributed proof system  $D[I]$  with a set of initial knowledge bases  $KB$ . Let  $\Phi$  be a finite sequence  $\langle \phi_1, \phi_2, \dots, \phi_n \rangle$  of formulas of the form  $(c \in KB_i)$  where  $c$  is a Datalog clause including quoted facts or a confidentiality policy<sup>3</sup>. We say that sequence  $\Phi$  is a proof for  $\phi_n$  if for each  $i$ ,  $1 \leq i \leq n$ , either

1.  $\phi_i$  holds in  $KB$ , or
2. There is an inference rule  $\frac{\Gamma}{\phi_i}$  in  $I$  such that  $\Gamma \subseteq \{\phi_1, \phi_2, \dots, \phi_{i-1}\}$ .

DEFINITION 4 ( $\vdash$ , PROVABLE). When there is a proof  $\Phi$  for a formula  $\phi$  in system  $D[I]$  with an initial set of knowledge bases  $KB$ , we say  $(I, KB) \vdash \phi$ , or  $KB \vdash \phi$  if a set of inference rules  $I$  is clear from the context.

In our reference system  $D[IS]$ , what is provable is equal to the fixpoint produced by the function  $fixpoint_i[IS]$ ; that is,  $KB \models (q \in KB_i)$  iff  $(IS, KB) \vdash (q \in KB_i)$ . In a confidentiality-preserving distributed proof system, confidentiality restrictions may prevent some facts from being provable. However, we require that every distributed proof system must satisfy the soundness requirement below.

DEFINITION 5 (SOUNDNESS). A distributed proof system  $D[I]$  is sound if  $\forall KB \in \mathcal{KB}^{Pl}, KB \models \phi$  if  $(I, KB) \vdash \phi$ .

In short, any fact that can be derived by a sound distributed proof system must also be derivable in the system  $D[IS]$ , which does not enforce any notion of access control on the facts in principals' knowledge bases.

### 3. SAFETY IN DISTRIBUTED PROVING

In this section, we define the notion of safety for distributed proof systems that support confidentiality policies to protect each principal's confidential facts from other unauthorized principals. We first describe an attacker model for distributed proof systems, and then define the safety of those systems based on the notion of nondeducibility.

#### 3.1 Attack Model

A distributed proof system  $D[I]$  consists of a finite set of principals  $P$ , and we consider attacks on a system from a finite set of malicious and colluding principals  $A \subset P$ . The malicious principals in  $A$  try to infer the truth of confidential facts maintained by non-malicious principals in  $P \setminus A$  by freely sharing information they learn through an execution of a system.

We represent the confidentiality policy as an access control list stored inside the knowledge base of each principal: a principal  $p_j$  is authorized to learn fact  $f$  from  $p_i$  if  $release(p_j, f) \in KB_i$ . Correspondingly, we say that a fact  $f$  in principal  $p_i$ 's knowledge base is confidential with respect to a set of principals  $A$  if there is not  $p_j$  such that  $release(p_j, f) \in KB_i$ . For notational convenience, we say a quoted fact  $(p_i \text{ says } f)$  is confidential if fact  $f$  at  $p_i$ 's knowledge base  $KB_i$  is confidential.

We assume that malicious principals in  $A$  have full control over their own knowledge bases; due to the use of the TTP model described in Section 2.1, they cannot actively affect the computation of a distributed proof system in any

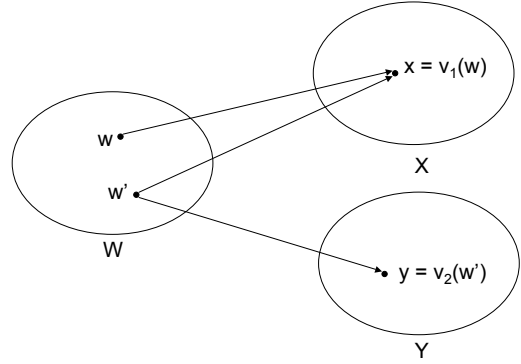


Figure 3: Concept of nondeducibility. For every world  $w \in W$  and every value  $y$  in  $Y$ , there must exist world  $w'$  such that  $v_1(w') = v_1(w)$  and  $v_2(w') = y$ .

other way. We also assume that the rules in each principal's knowledge base are private, and that there is no direct way for one principal to learn rules defined by other principals. However, each principal  $p_i$  may be able to learn whether a policy  $release(p_i, f)$  belongs to another principal  $p_j$ 's knowledge base  $KB_j$  (e.g., by requesting access to  $f$ ).

We assume that the sets of principals  $P$  and inference rules  $I$  parameterizing the system  $D$  are public knowledge. Thus, a malicious principal can compute the final state of a distributed proof system from any given (potentially guessed) initial configuration of the system. The malicious principals can iterate this process with different initial configurations frequently as they wish to perform inference attacks. We next give a formal definition of safety in a distributed proof system preventing inference attacks performed by malicious principals.

#### 3.2 Safety Definition

We develop a formal definition of safety for distributed proof systems based on the concept of *nondeducibility* introduced by Sutherland [15]. Sutherland modeled inferences by considering the set of all possible “worlds”  $\mathcal{W}$ —i.e., configurations of the system—and introduced the notion of an information function that represents a certain view of the system. That is, all the information about a given world  $w \in \mathcal{W}$  is provided as the outputs of information functions that operate on  $w$ .

Sutherland describes information flows from function  $v_1 : \mathcal{W} \rightarrow X$  to  $v_2 : \mathcal{W} \rightarrow Y$  as follows. For some world  $w$ , given  $x = v_1(w)$ , we can deduce that  $w$  belongs to a set of worlds  $S \subseteq \mathcal{W}$  where  $S = \{w' \mid w' \in \mathcal{W}, v_1(w') = x\}$ . If there is no world  $w' \in S$  such that  $v_2(w') = y$  for some  $y \in Y$ , then we can eliminate the possibility that  $v_2(w) = y$ , and we thus learn the information on  $v_2(w)$  from  $v_1(w)$ . Sutherland's nondeducibility in Definition 6 prohibits such information flow from function  $v_1$  to  $v_2$ , as illustrated in Figure 3.

DEFINITION 6 (NONDEDUCIBILITY). Given two information functions,  $v_1 : \mathcal{W} \rightarrow X$  and  $v_2 : \mathcal{W} \rightarrow Y$ , we say that no information flows from  $v_1$  to  $v_2$  if for every world  $w \in \mathcal{W}$ , and for  $y \in Y$ , there exists  $w' \in \mathcal{W}$  such that  $v_1(w) = v_1(w')$  and  $y = v_2(w')$ .

We apply this definition to a distributed proof system to define its safety with respect to a set of malicious principals

<sup>3</sup>We will introduce confidentiality policies in Section 3.1.

A described in Section 3.1. We define a set of worlds  $\mathcal{W}$ , and two information functions  $v_1$  and  $v_2$  in the following way. Given a distributed proof system  $D[I]$ , a world  $w \in \mathcal{W}$  corresponds to an initial set of the knowledge bases  $KB \in \mathcal{KB}^{\mathcal{P}}$ .

We define the function  $v_1$  to represent the knowledge that is learned by a set of malicious principals in  $A$  through an execution of system  $D[I]$ . We denote by  $KB^*$  a set of final knowledge bases that system  $D[I]$  computes from a given initial set of knowledge bases  $KB$ ; that is,

$$KB^* = \text{fixpoint}[I](KB).$$

We apply the same notation to the final knowledge bases  $KB^*$  and denote by  $KB_i^*$  a principal  $p_i$ 's final knowledge base. We now define the function  $v_1$ , which captures the knowledge of principals in set  $A$ , as follows:

**DEFINITION 7.** We define function  $v_1 : \mathcal{KB}^{\mathcal{P}} \times 2^{\mathcal{P}} \rightarrow \mathcal{KB}^{|\mathcal{A}|} \times \mathcal{KB}^{|\mathcal{A}|}$  such that:

$$v_1(KB, A) = \{(KB_i, KB_i^*) \mid p_i \in A\}.$$

The function  $v_1$  outputs a set of pairs of the initial and final set of knowledge bases maintained by the principals in set  $A$ . We assume that the pairs in the set are ordered based on their principal IDs.

Before introducing the function  $v_2$ , we define a set of confidential quoted facts in system  $D[I]$  as follows:

**DEFINITION 8.** We define the set of confidential quoted facts  $CF(KB)$  with respect to a set of principals  $A$  as follows:

$$CF(KB) = \{p_i \text{ says } f \mid f \in \mathcal{F} \wedge \forall p_j \in A : \text{release}(p_j, f) \notin KB_i\}.$$

The function  $v_2$  below outputs a set of confidential quoted facts that are *actually* stored in the knowledge bases  $KB$ .

**DEFINITION 9.** We define function  $v_2 : \mathcal{KB}^{\mathcal{P}} \rightarrow 2^{\mathcal{Q}}$  such that:

$$v_2(KB) = \{p_i \text{ says } f \mid (p_i \text{ says } f) \in CF(KB) \wedge f \in KB_i\}.$$

We now use the functions  $v_1$  and  $v_2$  to define the safety of a distributed proof system  $D[I]$  using the notion of non-deducibility from Definition 6.

**DEFINITION 10 (SAFETY).** We say that a distributed proof system  $D[I]$  is safe if for every set of knowledge bases  $KB \in \mathcal{KB}^{\mathcal{P}}$ , for every finite set of principals  $A \subset \mathcal{P}$ , for every finite set of quoted facts  $Q \subseteq CF(KB)$ , there exists another set of knowledge bases  $KB'$  such that:

1.  $v_1(KB) = v_1(KB')$ , and
2.  $Q = v_2(KB'^*)$ .

The first condition above requires a set of malicious principals  $A$  to obtain the same initial and final states regardless of with which initial state the system starts. The second condition requires that an alternate configuration could have any subset of confidential quoted facts in the final states of the alternate system. Note that we consider protecting the truth of confidential facts in the final state because it implies protecting their truth in the initial state as well.

In the next section, we consider several different distributed proof systems and show how we analyze the safety of those systems based on the safety in Definition 10.

## 4. SAFETY ANALYSIS

We now use our framework to analyze several example confidentiality-preserving distributed proof systems to determine whether they respect the safety property described in Definition 10. We first examine the DAC system, which enforces confidentiality policies in a pairwise. We next consider two proof systems that support inferences on encrypted logical statements. The Nested Encryption (NE) system models the system defined by Minami and Kotz [14] that uses public-key encryption to encrypt conjunctions of truth values in a nested fashion. We then discuss the Commutative Encryption (CE) system, which uses a commutative encryption scheme where a value can be encrypted to multiple principals who can decrypt it in arbitrary order. We show that the DAC and NE systems are safe, whereas the CE system is not safe despite supporting a seemingly-reasonable cryptographic construction.

### 4.1 DAC System $D[I_{DAC}]$

The DAC system enforces each principal's confidentiality policies in the most intuitive way; the system requires that each principal  $p_i$ 's facts are released only to principals who satisfy  $p_i$ 's confidentiality policies. The DAC system derives new facts and quoted facts with the inference rules in Figure 4.

$$\begin{aligned} \text{(COND)} \quad & \frac{(f \leftarrow q_1, \dots, q_n) \in KB_i \quad q_k \in KB_i \text{ for all } k}{f \in KB_i} \\ \text{(DAC-SAYS)} \quad & \frac{f \in KB_i \quad \text{release}(p_j, f) \in KB_i}{(p_i \text{ says } f) \in KB_i} \end{aligned}$$

**Figure 4: Inference rules  $I_{DAC}$  of the DAC system.**

The DAC system uses the same (COND) rule of the reference system  $D[I_S]$  to allow each principal to derive local facts. However, the (DAC-SAYS) rule, which replaces the (SAYS) rule in  $I_S$ , requires that  $p_i$  maintains a DAC policy  $\text{release}(p_j, f)$  in its knowledge base  $KB_i$  in order for another principal  $p_j$  to receive quoted fact  $(p_i \text{ says } f)$ .

We first show the soundness result of the DAC system.

**PROPOSITION 1.** The DAC system is sound.

**PROOF.** Suppose that the DAC system derives a formula  $\phi$  with a proof  $\Phi$ . Let  $\Phi'$  be a reduced version of the sequence  $\Phi$  with all of the *release* formulas removed. Then  $\Phi'$  is a proof for  $\phi$  in the reference proof system  $D[I_S]$ : any intermediate  $\phi_i$  in  $\Phi$  that is the result of the (COND) rule can be derived by the same rule in  $D[I_S]$ , and any  $\phi_i$  that is the result of the (DAC-SAYS) rule can be derived by the (SAYS) rule in  $D[I_S]$ . Thus, the DAC system satisfies the soundness requirement in Definition 5.  $\square$

We next prove that the DAC system is safe. We first provide an intuition for why system  $D_{DAC}$  is safe. The main observation is that a malicious principal  $p_n$  in  $A$  never directly receives a confidential quoted fact  $(p_i \text{ says } f') \in CF(KB)$ . However, it is possible that  $p_n$  receives a non-confidential quoted fact  $(p_i \text{ says } f)$  derived from  $f'$  in  $KB_i$ , as demonstrated in Figure 5(a). When principal  $p_n$  derives this quoted fact,  $p_n$  does not know that the sender principal

$$\text{(COND)} \frac{(f \leftarrow f') \in KB_i}{f' \in KB_i}$$

$$\text{(DAC-SAYS)} \frac{f \in KB_i \quad \text{release}(p_n, f) \in KB_i}{(p_i \text{ says } f) \in KB_n}$$

(a) Original proof

$$\text{(DAC-SAYS)} \frac{f \in KB_i \quad \text{release}(p_n, f) \in KB_i}{(p_i \text{ says } f) \in KB_n}$$

(b) Alternate proof

**Figure 5: An example of two proofs indistinguishable from a principal  $p_n$ .**

$p_i$  derives fact  $f$  using rule  $f \leftarrow f'$  because that rule is private from the other principals. Therefore,  $p_n$  consider the alternate proof shown in Figure 5(b) also possible. This observation provides the intuition of the proof of the following theorem:

We now show that DAC system satisfies the safety conditions in Definition 10.

**THEOREM 1.** *The DAC system  $D[I_{DAC}]$  is safe.*

**PROOF.** Given a set of initial knowledge bases  $KB$  and a set of confidential quoted facts  $Q$ , and a set of malicious principals  $A$ , Algorithm 1 computes an alternate set of knowledge bases  $KB'$  satisfying the safety conditions in Definition 10. Lines 4–6 ensure that every malicious principal

**Algorithm 1** Compute an alternate configuration  $KB'$  in the DAC system.

---

```

1: % INPUT:  $KB \in \mathcal{KB}, Q \in 2^Q, A \in 2^P$ 
2: % OUTPUT:  $KB' \in \mathcal{KB}$ 
3:
4: for all  $p_n \in A$  do
5:    $KB'_n \leftarrow KB_n$ 
6: end for
7: for all  $p_i \in (P \setminus A)$  do
8:    $KB'_i \leftarrow \{\text{release}(p_j, f) \mid \text{release}(p_j, f) \in KB_i\}$ 
9: end for
10: for all  $(p_i \text{ says } f) \in Q$  do
11:   if  $(p_i \text{ says } f) \notin CF(KB)$  then
12:     for all  $p_n \in A$  such that  $(p_i \text{ says } f) \in KB_n^*$  do
13:        $KB'_i \leftarrow KB'_i \cup \{f\}$ 
14:     end for
15:   else if  $(p_i \text{ says } f) \in Q$  then
16:      $KB'_i \leftarrow KB'_i \cup \{f\}$ 
17:   end if
18: end for

```

---

$p_n$  has an initial knowledge base  $KB'_n$ , which is same as that in the alternate systems. Lines 7–9 initialize each non-malicious principal  $p_i$ 's knowledge base  $KB'_i$  to an empty set of facts while maintaining the original release policies. Lines 10–14 ensure that, if a malicious principal  $p_n$  derives a quoted fact  $(p_i \text{ says } f)$  in the original  $KB_n^*$ ,  $p_n$  derives the same quoted fact in the alternate system by adding fact  $f$  into  $p_i$ 's alternate knowledge base  $KB'_i$ . Since no non-malicious principal maintains any rule for deriving new facts in the alternate system,  $p_n$  does not derive any additional quoted facts in the alternate system. Therefore, every malicious principal obtains the same knowledge base as in the

$$\text{(ECOND)} \frac{(f \leftarrow q_1, \dots, q_n) \in KB_i \quad (q_k, e_k) \in KB_i \text{ for all } k}{\left(f, \bigwedge_{k=1}^n e_k\right) \in KB_i}$$

$$\text{(DEC1)} \frac{(q, E_i(e) \wedge e') \in KB_i}{(q, e \wedge e') \in KB_i} \quad \text{(DEC2)} \frac{(q, \text{True}) \in KB_i}{q \in KB_i}$$

$$\text{(ENC-SAYS)} \frac{(f, e) \in KB_i \quad \text{release}(p_j, f) \in KB_i}{(p_i \text{ says } f, E_j(e)) \in KB_k}$$

**Figure 6: Inference rules  $I_{NE}$  of the NE system.**

original system. Lines 15–17 add fact  $f$  into  $p_i$ 's alternate knowledge base  $KB'_i$  if quoted fact  $(p_i \text{ says } f)$  belongs to a given set of quoted facts  $Q$ . Since every non-malicious principal  $p_i$  does not derive any new facts,  $p_i$  maintains a confidential fact  $f$  in  $KB'_i^*$  if and only if  $(p_i \text{ says } f) \in Q$ . Since  $KB'$  computed with Algorithm 1 satisfies the two conditions described in Definition 10, the DAC system is safe.  $\square$

## 4.2 NE System $D[I_{NE}]$

We next consider the NE system, which abstracts the cryptographic proving protocol developed by Minami and Kotz in [14]. The NE system represents public-key operations by which logical statements (facts or quoted facts) are encrypted in a nested way. In this section, we show that the NE system is sound and safe and that it can derive more facts than the DAC system. The inference rules of the NE system model public key operations performed by the principals of the system, each of which maintains a public/private key pair and also knows other principals' public keys.

In the NE system, we encode a quoted fact in an encrypted form as a tuple  $(q, e)$ , where  $q$  is a quoted fact and  $e$  is a ciphertext containing the truth (true or false) of the quoted fact  $q$ . We say that a quoted fact  $q$  ( $\equiv p_i \text{ says } f$ ) is true if  $f \in KB_i$ .<sup>4</sup> An encrypted value  $e$  is represented by the following grammar in BNF:

$$e ::= \text{True} \mid \text{False} \mid E_i(e) \mid e \wedge e,$$

where a term  $E_i(e)$  represents a data object in which value  $e$  is encrypted with principal  $p_i$ 's public key. The grammar shows that the truth of a quoted fact can be encrypted recursively with different principals' keys. For example, we represent a value True encrypted with a principal  $p_i$ 's public key first and with  $p_j$ 's public key next as  $E_j(E_i(\text{True}))$ . Also, it can be the conjunction of multiple encrypted values (e.g.,  $E_i(\text{True}) \wedge E_j(\text{True})$ ).

We now describe the set of inference rules  $I_{NE}$  for the NE system listed in Figure 6. The inference rule (ECOND) is similar to the rule (COND) in the DAC system, but it applies a rule to a set of encrypted facts of the form  $(q_i, e_i)$ , and derives a new encrypted fact  $(f, \bigwedge_{k=1}^n e_k)$ ; that is, a quoted fact  $q$  is true if all the truth values in  $e_1, \dots, e_n$  are true. The inference rule (DEC1) represents a principal  $p_i$ 's decryption

<sup>4</sup> We can consider that such a tuple is a special form of an atom where we omit a predicate symbol from the ordinary Datalog atom (e.g.,  $enc(q, e)$ ).

operation on an encrypted fact. The rule allows principal  $p_i$  to remove the outermost encryption of  $E_i(e)$  in conjunction  $E_i(e) \wedge e'$  and produces a new conjunction  $e \wedge e'$ . The inference rule (DEC2) derives an ordinary quoted fact  $q$  from a tuple whose  $e$  component is just True. The inference rule (ENC-SAYS) allows a principal  $p_i$  maintaining an encrypted fact  $(f, e)$  to export  $(p_i \text{ says } f, E_j(e))$  to any principal  $p_k$  if  $p_i$  maintains a confidentiality policy  $\text{release}(p_j, f)$ .

*Example.* Consider the NE system with the following initial set of knowledge bases  $KB$ .

$$\begin{aligned} KB_3 &= \emptyset, \\ KB_2 &= \{f_2 \leftarrow p_0 \text{ says } f_0, p_1 \text{ says } f_1, \text{release}(p_3, f_2)\}, \\ KB_1 &= \{f_1, \text{release}(p_3, f_1)\}, \\ KB_0 &= \{f_0, \text{release}(p_2, f_0)\}. \end{aligned}$$

Figure 7 shows a proof for  $p_2 \text{ says } f_2 \in KB_3$ . Notice that principal  $p_2$ , which is not authorized to learn the truth of fact  $f_1$  in  $KB_1$  is involved in the construction of the proof. Notice that principal  $p_3$  applies the rule (DEC1) twice to decrypt the encrypted value  $E_3(E_3(\text{True}))$ , which is encrypted by principals  $p_1$  and  $p_2$ .

we show the soundness result of the NE system below.

PROPOSITION 2. *The NE System  $D[INE]$  is sound.*

PROOF. Suppose that  $(INE, KB) \vdash \phi$  with a proof  $\Phi$ . We can obtain a proof  $\Phi'$  for the system  $D[IS]$  by first removing from  $\Phi$  formulas regarding confidentiality policies and then replacing every tuple  $(q, e)$  in  $\Phi$  where  $e$  contains a value True with  $q$ . Note that this effectively converts the inference rules (ECOND) and (ENC-SAYS) into (COND) and (SAYS), respectively, and eliminates the applications of the rules (DEC1) and (DEC2).  $\square$

We next show that the NE system is safe. We assume that the TTP for the NE system discards all the encrypted facts from the final state of a system. Our approach for our safety proof is similar to that for the DAC system in the sense that if a malicious principal  $p_n$  derives a quoted fact  $(p_i \text{ says } f)$ , then we try to make an alternate indistinguishable proof for  $((p_i \text{ says } f) \in KB_n)$ , which does not depend on non-malicious principals' confidential facts. In the case of the DAC system, we construct a proof with a single formula  $(f \in KB_i)$  by adding fact  $f$  into  $p_i$ 's alternate initial knowledge base  $KB'_i$ . However, we cannot use this simple strategy because a non-malicious principal  $p_i$  in the NE system might export a quoted fact whose truth value is recursively encrypted with the public keys of malicious principals. Suppose that  $p_i$  exports an encrypted quoted fact  $(p_i \text{ says } f, E_j(E_k(\text{True})))$  where  $p_j$  and  $p_k$  are malicious principals in set  $A$ . The encrypted value in the quoted fact must be first decrypted by principal  $p_j$  and then by  $p_k$ . In this process, only  $p_k$  derives some quoted fact in cleartext. However, if we add fact  $f$  into  $p_i$ 's knowledge base, principal  $p_j$  derives  $(p_i \text{ says } f)$  in  $KB_j$ , which does not exist in the original system. Therefore, we need to make sure that an alternate proof constructs an encrypted value of the same structure.

We now prove that the NE system is safe by giving a precise procedure for constructing an alternate configuration  $KB'$  in which the malicious principals derive the exact same set of encrypted quoted facts regardless of the state of con-

fidential facts in the knowledge bases of honest principals.

THEOREM 2. *The NE system  $D[INE]$  is safe.*

PROOF (SKETCH). Given a set of initial knowledge bases  $KB$ , a set of confidential quoted facts  $Q$ , and a set of malicious principals  $A$ , Algorithm 2 computes an alternate set of knowledge bases  $KB'$  satisfying the safety conditions in Definition 10. This algorithm consists of two parts: the first part converts every proof  $\Phi$  for  $(p_i \text{ says } f) \in KB_n$  where  $p_i \notin A$  and  $p_n \in A$  into  $\Phi'$  by modifying only the rules of non-malicious principals. We ensure that if a proof  $\Phi$  contains confidential quoted facts, they only appear in the bodies of non-malicious principals' rules in  $\Phi'$ . This property is crucial to eliminate the proof  $\Phi$ 's dependencies on non-malicious principals' confidential facts in the alternate system. The second part further modifies non-malicious principals' rules as we add or remove a confidential quoted fact to satisfy the requirement regarding set  $Q$ . We construct an alternate proof  $\Phi'$  equivalent to  $\Phi$  without introducing any proof that allows a malicious principal to derive quoted facts that do not exist in the original configuration.

We now describe Algorithm 2 in detail. Line 1 initially sets an alternate set of knowledge bases  $KB'$  to be same as the original  $KB$ . Lines 2–7 iteratively combine the chains of rules that could appear in a proof  $\Phi$  for  $q \in KB_n$  where  $q$  is a quoted fact and  $p_n \in A$ , in the original system. Suppose that  $\Phi$  contains the following subsequence  $\Gamma$  omitting some formulas in  $\Phi$ .

$$\begin{aligned} &< f_j \leftarrow q'_1, \dots, q'_p \in KB_j, \\ &\text{release}(p_k, f_j) \in KB_j, \\ &(f_k \leftarrow q_1, \dots, q_m, p_j \text{ says } f_j) \in KB_k > \end{aligned}$$

where  $p_k \notin A$ . If we combine the two rules in  $\Gamma$  by replacing quoted fact  $(p_j \text{ says } f_j)$  in the body of the rule in  $KB_k$  with the set of quoted facts  $q'_1, \dots, q'_p$  in the body of the rule in  $KB_j$ , we obtain another subsequence  $\Gamma'$  below.

$$< (f_k \leftarrow q_1, \dots, q_m, q'_1, \dots, q'_p) \in KB_k >$$

We can obtain an alternate proof  $\Phi'$  by replacing subsequence  $\Gamma$  in  $\Phi$  with  $\Gamma'$  because, in proof  $\Phi$ , the encryption operation on quoted fact  $(p_j \text{ says } f_j)$  with principal  $p_k$ 's public key is immediately followed by the decryption operation with  $p_k$ 's private key. Thus, these two operations do not leave any effect on the structure of any encrypted value that appears in the formulas following  $\Gamma'$  in  $\Phi'$ .

By iterating the operations of modifying non-malicious principals' rule in the while loop, we eventually obtain an alternate proof  $\Phi'$  in which every confidential quoted fact of the form  $(p_i \text{ says } f)$  only appears in the bodies of rules maintained by non-malicious principals satisfying  $p_i$ 's confidentiality policy on fact  $f$ ; if a principal  $p_j$  maintains such a rule,  $p_i$  should have the policy  $\text{release}(p_j, f)$  in  $KB_i$ .

The main observation is that, if a non-malicious principal  $p_i$  exports an encrypted fact  $(p_i \text{ says } f, E_k(e))$  in a proof  $\Phi$  for  $q \in KB_n$  where  $p_n$  is a malicious principal, another non-malicious principal  $p_k$  must perform a decryption operation on  $E_k(e)$  before  $p_n$  derives quoted fact  $q$  in  $KB_n$ ; otherwise,  $p_n$  cannot decrypt encrypted quoted fact  $(q, e)$  where  $e$  contains  $E_k(e)$ , which is originally produced by  $p_i$ . This requirement further requires that, if the encrypted value  $E_k(e)$  is further encrypted with different principals' public keys recursively, the principals possessing the corresponding private

(ENC-SAYS)	$\frac{f_0 \in KB_0 \quad \text{release}(p_2, f_0) \in KB_0}{(p_0 \text{ says } f_0, E_2(\text{True})) \in KB_2}$	$\frac{f_1 \in KB_1 \quad \text{release}(p_3, f_1) \in KB_0}{(p_1 \text{ says } f_1, E_3(\text{True})) \in KB_2}$	(ENC-SAYS)
(ECOND)	$(f_2 \leftarrow p_0 \text{ says } f_0, p_1 \text{ says } f_1) \in KB_2$		
(DEC1)	$(f_2, E_2(\text{True}) \wedge E_3(\text{True})) \in KB_2$		
(ENC-SAYS)	$(f_2, E_3(\text{True})) \in KB_2$		release( $p_3, f_2$ ) $\in KB_2$
(DEC1)	$(p_2 \text{ says } f_2, E_3(E_3(\text{True}))) \in KB_3$		
(DEC1)	$(p_2 \text{ says } f_2, E_3(\text{True})) \in KB_3$		
(DEC2)	$(p_2 \text{ says } f_2, \text{True}) \in KB_3$		
	$p_2 \text{ says } f_2 \in KB_3$		

Figure 7: Example proof  $\Phi$  in the NE system.

keys must perform decryption operations in the reverse order so that  $p_k$  can receive the encrypted value  $E_k(e)$  and decrypt it. This condition allows us to keep finding a matching pair of two rules to be combined in Lines 2–3 of the algorithm until we only have a non-malicious  $p_k$ 's rule whose body contains quoted facts of other non-malicious principals.

Lines 8–24 add or remove confidential facts depending on whether they belong to set  $Q$  or not. Lines 9–15 cover the case where we remove confidential facts that are derived in the original system but do not belong to set  $Q$ . Line 10 removes fact  $f$  and all rules for deriving  $f$  from  $p_i$ 's knowledge base  $KB'_i$ . Lines 12–13 remove quoted fact  $(p_i \text{ says } f)$  from the body of any rule including that quoted fact as one of the conditions. This operation ensures that a proof  $\Phi$  in the original system, which relies on fact  $f$  in  $KB_i$ , is not destroyed in the alternate system; we convert  $\Phi$  into another proof  $\Phi'$  without using formula  $(f \in KB_i)$ . As we mention earlier, only non-malicious principals maintain such rules, and thus we only modify non-malicious principals' knowledge bases. Lines 16–23 cover the case where we add confidential facts that are not derived in the original system, but are included in set  $Q$ . Line 17 adds fact  $f$  into  $p_i$ 's knowledge base  $KB'_i$ , and lines 18–22 remove all the rules that include quoted fact  $(p_i \text{ says } f)$  in its body such that formula  $(f \in KB'_i)$  never be part of a proof for deriving new facts in the alternate system. Since Algorithm 2 ensures that every malicious principal  $p_n$  derives the same set of encrypted quoted facts in the alternate system by modifying only non-malicious principals' knowledge bases, we conclude that the NE system is safe.  $\square$

### 4.3 CE System $D[I_{CE}]$

The CE system  $D[I_{CE}]$  extends the NE system by supporting commutative encryption, in which a value encrypted using multiple, different public keys in some order can be decrypted using the corresponding private keys in *any* order. We represent a quoted fact encrypted with a commutative encryption scheme as a tuple  $(q, S)$  where set  $S$  contains a set of principals whose public keys are used to encrypt a boolean value True.

Figure 8 shows the set of inference rules describing the System  $D_{CE}$ , which are obtained by modifying inference rules for the NE System (Figure 6) to support the commutative encryption. The inference rule (CECOND) corresponds to the inference rule (ECOND) in the NE System, but the difference is that the derived fact  $f$  is associated with the

**Algorithm 2** Compute an alternate configuration  $KB'$  in the NE system.

---

```

1:  $KB' \leftarrow KB$ 
2: while  $\exists(r \equiv f_k \leftarrow q_1, \dots, q_m, p_j \text{ says } f_j) \in KB'_k$  such that
    $p_k \notin A$  and  $\exists f_j \leftarrow q'_1, \dots, q'_p \in KB'_j$  do
3:    $KB'_k \leftarrow KB'_k \setminus \{r\}$ 
4:   for all  $(f_j \leftarrow q'_1, \dots, q'_p) \in KB'_j$  such that  $\text{release}(p_k, f_j) \in$ 
      $KB'_j$  do
5:      $KB'_k = KB'_k \cup \{f_k \leftarrow q_1, \dots, q_m, q'_1, \dots, q'_p\}$ 
6:   end for
7: end while
8: for all  $q \equiv (p_i \text{ says } f)$  such that  $q \in CF(KB)$  do
9:   if  $q \notin Q$  then
10:     $KB'_i \leftarrow KB'_i \setminus \{f\} \setminus \{r \mid r \in KB'_i \wedge \text{head}(r) = f\}$ 
11:    for all  $(r \equiv f_k \leftarrow q_1, \dots, q_m, p_i \text{ says } f_i)$  in  $KB_k$  such that
       $p_k \notin A$  do
12:      if  $f \in KB'_i$  then
13:         $KB'_k \leftarrow KB'_k \setminus \{r\} \cup \{f_k \leftarrow q_1, \dots, q_m\}$ 
14:      end if
15:    end for
16:   else
17:     $KB'_i \leftarrow KB'_i \cup \{f\}$ 
18:    for all  $(r \equiv f_k \leftarrow q_1, \dots, q_m, p_i \text{ says } f_i)$  in  $KB_k$  such that
       $p_k \notin A$  do
19:      if  $f \notin KB'_i$  then
20:         $KB'_k \leftarrow KB'_k \setminus \{r\}$ 
21:      end if
22:    end for
23:   end if
24: end for

```

---

union of all the principals' public keys needed to decrypt quoted facts  $q_1, \dots, q_n$ . The inference rule (CEDEC) allows  $p_i$  to partially decrypt an encrypted quoted fact  $(q, S)$  with  $p_i$ 's public key and obtain  $(q, S \setminus \{i\})$ . If  $S = \{i\}$ , then the quoted fact  $q$  is derived into  $KB_i$ . The inference rule (CE-SAYS) allows principal  $p_i$  maintaining an encrypted fact  $(f, S)$  to export  $(q, S \cup \{j\})$  to any principal  $p_k$  if  $p_i$  maintains a DEC policy  $\text{release}(p_j, f)$ .

The commutative encryption in the CE system is less restrictive than the nested encryption used in the NE system, and, therefore, system  $D_{CE}$  allows principals in the system to derive more facts than the NE system does. Unfortunately, the CE system is *not* safe as we show below.

**THEOREM 3.** *The CE system  $D[I_{CE}]$  is unsafe.*

**PROOF.** We now show that the CE system is not safe by giving an example of unsafe derivations. Consider the

$$\begin{aligned}
& \frac{(f \leftarrow q_1, \dots, q_n) \in KB_i}{(q_k, S_k) \in KB_k \text{ for all } k} \\
\text{(CECOND)} & \frac{}{(f, \cup_{k=1}^n S_k) \in KB_i} \\
& \frac{(q, S) \in KB_i}{(q, (S \setminus \{p_i\})) \in KB_i} \\
\text{(CEDEC)} & \frac{}{} \\
& \frac{(f, S) \in KB_i \quad \text{release}(p_j, f) \in KB_i}{(p_i \text{ says } f, (S \cup \{p_j\})) \in KB_k} \\
\text{(CE-SAYS)} & \frac{}{}
\end{aligned}$$

**Figure 8: Inference rules  $I_{CE}$  of the CE system**

following initial set of knowledge bases  $KB$ .

$$\begin{aligned}
KB_4 &= \emptyset, \\
KB_3 &= \{f_3 \leftarrow p_2 \text{ says } f_2, \text{release}(p_4, f_3)\}, \\
KB_2 &= \{f_2 \leftarrow p_1 \text{ says } f_1, \text{release}(p_3, f_2)\}, \\
KB_1 &= \{f_1 \leftarrow p_0 \text{ says } f_0, \text{release}(p_4, f_1)\}, \\
KB_0 &= \{f_0, \text{release}(p_2, f_0)\}.
\end{aligned}$$

Figure 9 shows a proof  $\Phi$  for  $(p_3 \text{ says } f_3) \in KB_4$  in the CE system. Suppose that principals  $p_1$ ,  $p_3$ , and  $p_4$  are malicious and in set  $A$  and that fact  $f_0$  in principal  $p_0$ 's knowledge base  $KB_0$  is confidential with respect to the principals in  $A$ . There are two possible ways to construct an alternate proof  $\Phi'$  without using confidential fact  $f_0$  in  $KB_0$ . One way is to add fact  $f_2$  into  $KB_2$ . However, this addition allows malicious principal  $p_3$  to derive quoted fact  $(p_2 \text{ says } f_2)$  in  $KB_3$  in  $\Phi'$ , which does not exist in the original system. If we replace  $p_2$ 's confidentiality policy with a new policy  $\text{release}(p_4, p_0)$ , principal  $p_4$  instead derives a quoted fact  $(p_2 \text{ says } f_2)$  in  $KB_4$ , and thus the alternate proof is still distinguishable. The other way is to arrange the following alternate knowledge bases of  $p_0$  and  $p_2$  so that  $p_2$  exports an quoted fact encrypted with  $p_0$ 's public key as in the original proof.

$$\begin{aligned}
KB'_2 &= \{f_2 \leftarrow p_0 \text{ says } f'_0, \text{release}(p_3, f_2)\}, \\
KB'_0 &= \{f'_0, \text{release}(p_4, f'_0)\}.
\end{aligned}$$

However,  $p_4$  derives a new quoted fact  $(p_0 \text{ says } f'_0)$  in  $KB_4$ , and thus the resulting alternate proof is again distinguishable. Since there is no way to construct an indistinguishable alternate proof  $\Phi'$ , we conclude that the CE system is not safe.  $\square$

## 5. RELATED WORK

Distributed proof systems have been studied mainly in the context of distributed authorization systems. Many researchers [1, 2, 4, 6, 8, 9, 11, 13] have proposed new logic-based languages to express security policies based on roles, delegations, and so on. On the other hand, our focus is to study different proof theories of Datalog-based distributed proof systems concerning the confidentiality of logical statements in each principal's knowledge base. Although a few researchers [14, 17, 18] proposed a confidentiality-preserving distributed proof system in which each peer's rules and facts can be private, none of them provides a formal definition of confidentiality considering inferences attacks.

Becker's information flow analysis [3] on the issue of unauthorized inferences in logic-based authorization systems is

the closest to our research. However, Becker's security model is weaker than ours in a couple of ways. First, Becker's safety definition is based on the notion of opacity, which ensures that a querier cannot determine the truth of each confidential fact. Our definition, on the other hand, requires that every possible truth assignment to confidential facts is possible from the viewpoint of an adversary. Second, a querier in Becker's model does not participate in the process of constructing a proof for an initial query although the querier is allowed to insert additional policies into the system. An adversary in our model is a set of colluding principals, which participate in the process of constructing proofs. The malicious principals that are not an initial querier can gain knowledge about intermediate results of the proof.

We previously developed a confidentiality-preserving distributed protocol [10] that allows a querier to derive the conjunction of multiple facts in a distributed proof system. Confidentiality policies in that system are more general than those in the current paper in the sense that the release of a fact's truth value can be made contingent upon facts managed by other principals. However, the safety analysis of the protocol based on the TTP model only considers a single construction of a proof for a given query.

Winsborough and Li [16] formally define safety in automatic trust negotiation based on the notion of *indistinguishability* about the possession of a set of credentials by a negotiating party. Their analysis considers a sequence of messages between *two* parties while distributed proof systems in this paper could involve *more than* two principals.

## 6. CONCLUSION

We study the safety of distributed proof systems in which each principal of the system protects its logical statements with confidentiality policies. We model distributed proof systems as a set of inference rules controlling the conditions under which facts can be exchanged between disparate knowledge bases. Our safety definition based on the notion of nondeducibility [15] ensures that any give set of malicious principals in the system cannot gain any information on the truth assignment of confidential facts maintained by non-malicious principals.

Our safety analysis shows that the NE system, whose inference rules encode ordinary public-key operations, is safe and proves more facts than the DAC system, which locally enforces confidentiality policies based on two-party communication. Our theoretical framework for safety analysis is effective enough to prove that the CE system, which supports commutative encryption, is unsafe despite supporting a seemingly-useful cryptographic fact derivation scheme.

*Acknowledgments.* This work was supported in part by National Science Foundation grants CNS 07-16421, CCF-0916015, and CNS-1017229; a grant from Motorola Mobile Devices; and a grant from the Promotion program for Reducing global Environmental load through ICT innovation (PREDICT) of the Ministry of Internal Affairs and Communications in Japan.

## 7. REFERENCES

- [1] Andrew W. Appel and Edward W. Felten. Proof-carrying Authentication. In *Proceedings of the 6th ACM Conference on Computer and*

(CE-SAYS)	$\frac{(f_0 \in KB_0) \quad release(p_2, f_0) \in KB_0}{(p_0 \text{ says } f_0, \{p_2\}) \in KB_1}$	$(f_1 \leftarrow p_0 \text{ says } f_0) \in KB_1$	
(CECOND)			$release(p_4, f_1) \in KB_1$
(CE-SAYS)	$\frac{(f_1, \{p_2\}) \in KB_1}{(p_1 \text{ says } f_1, \{p_2, p_4\}) \in KB_2}$	$(f_2 \leftarrow p_1 \text{ says } f_1) \in KB_2$	
(CECOND)			
(CEDEC)	$(f_2, \{p_2, p_4\}) \in KB_2$		
(CE-SAYS)	$\frac{(f_2, \{p_4\}) \in KB_2 \quad release(p_3, f_2) \in KB_2}{(p_2 \text{ says } f_2, \{p_3, p_4\}) \in KB_3}$	$(f_3 \leftarrow p_2 \text{ says } f_2) \in KB_3$	
(CECOND)			
(CEDEC)	$(f_3, \{p_3, p_4\}) \in KB_3$		
(CE-SAYS)	$\frac{(f_3, \{p_4\}) \in KB_3 \quad release(p_4, f_3) \in KB_3}{(p_3 \text{ says } f_3, \{p_4\}) \in KB_4}$		
(CEDEC)		$(p_3 \text{ says } f_3) \in KB_4$	

**Figure 9: Example unsafe derivations in the CE system.**

- Communications Security*, pages 52–62. ACM Press, 1999.
- [2] Lujo Bauer, Scott Garriss, and Michael K. Reiter. Distributed Proving in Access-Control Systems. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 81–95, Washington, DC, USA, 2005. IEEE Computer Society.
- [3] Moritz Y. Becker. Information flow in credential systems. *Computer Security Foundations Symposium, IEEE*, 0:171–185, 2010.
- [4] Moritz Y. Becker, Cedric Fournet, and Andrew D. Gordon. Design and semantics of a decentralized authorization language. *Proceedings of the 20th IEEE Computer Security Foundations Symposium*, pages 3–15, July 2007.
- [5] Stefano Ceri, Georg Gottlob, and Letizia Tanca. What you always wanted to know about datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering*, 1(1):146–166, March 1989.
- [6] John DeTreville. Binder, a logic-based security language. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, page 105, Washington, DC, USA, 2002. IEEE Computer Society.
- [7] Oded Goldreich. *Foundations of Cryptography Volume 2: Basic Applications*. Cambridge University Press, 2004.
- [8] Yuri Gurevich and Itay Neeman. Dkal: Distributed-knowledge authorization language. *Proceedings of the 21st IEEE Computer Security Foundations Symposium*, pages 149–162, June 2008.
- [9] Trevor Jim. SD3: A trust management system with certified evaluation. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, pages 106–115. IEEE Computer Society, 2001.
- [10] Adam J. Lee, Kazuhiro Minami, and Nikita Borisov. Confidentiality-preserving distributed proofs of conjunctive queries. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 287–297, New York, NY, USA, 2009. ACM.
- [11] Ninghui Li, Joan Feigenbaum, and Benjamin N. Grosf. A logic-based knowledge representation for authorization with delegation. In *Proceedings of the 1999 IEEE Computer Security Foundations Workshop*, page 162, Washington, DC, USA, 1999. IEEE Computer Society.
- [12] Ninghui Li, Benjamin N. Grosf, and Joan Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM Transactions on Information and System Security*, 6(1):128–171, 2003.
- [13] Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a role-based trust-management framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, page 114, Washington, DC, USA, 2002. IEEE Computer Society.
- [14] Kazuhiro Minami and David Kotz. Secure context-sensitive authorization. *Journal of Pervasive and Mobile Computing*, 1(1):123–156, March 2005.
- [15] David Sutherland. A model of information. In *9th National Computer Security Conference*, pages 175–183, September 1986.
- [16] William H. Winsborough and Ninghui Li. Safety in automated trust negotiation. In *Proceedings of the 2004 IEEE Symposium on Security and Privacy*, pages 147–160. IEEE Computer Society, May 2004.
- [17] Marianne Winslett, Charles C. Zhang, and Piero A. Bonatti. PeerAccess: a logic for distributed authorization. In *Proceedings of the 12th ACM Conference on Computer and Communications Security*, pages 168–179, New York, NY, USA, 2005. ACM Press.
- [18] Charles C. Zhang and Marianne Winslett. Distributed authorization by multiparty trust negotiation. In *Proceedings of the 13th European Symposium on Research in Computer Security*, pages 282–299, Berlin, Heidelberg, 2008. Springer-Verlag.