Security Analysis of Cryptographically Controlled Access to XML Documents

MARTÍN ABADI
University of California, Santa Cruz, and Microsoft Research, Silicon Valley, USA
and
BOGDAN WARINSCHI
University of Bristol, UK

Some promising recent schemes for XML access control employ encryption for implementing security policies on published data, avoiding data duplication. In this paper we study one such scheme, due to Miklau and Suciu. That scheme was introduced with some intuitive explanations and goals, but without precise definitions and guarantees for the use of cryptography (specifically, symmetric encryption and secret sharing). We bridge this gap in the present work. We analyze the scheme in the context of the rigorous models of modern cryptography. We obtain formal results in simple, symbolic terms close to the vocabulary of Miklau and Suciu. We also obtain more detailed computational results that establish security against probabilistic polynomial-time adversaries. Our approach, which relates these two layers of the analysis, continues a recent thrust in security research and may be applicable to a broad class of systems that rely on cryptographic data protection.

Categories and Subject Descriptors: E.3 [**Data Encryption**]: ; H.2.7 [**Database Administration**]: Security, integrity, and protection; F.1.1 [**Models of Computation**]: Relations between models

General Terms: Security, Theory

Additional Key Words and Phrases: access control, authorization, encryption, XML

1. INTRODUCTION

A classic method for enforcing policies on access to data is to keep all data in trusted servers and to rely on these servers for mediating all requests by clients, authenticating the clients and performing any necessary checks. An alternative method, which is sometimes more attractive, consists in publishing the data in such a way that each client can see only the appropriate parts. In a naive scheme, many sani-

This work was supported in part by the National Science Foundation under Grants CCR-0204162, CCR-0208800, CCF-0524078, and ITR-0430594, and by ACI Jeunes Chercheurs JC 9005 and ARA SSIA Formacrypt.
It was partly carried out while Bogdan Warinschi was affiliated with the University of California at Santa Cruz, with Stanford University, and with Loria, INRIA, in Nancy.
Authors's addresses: Martín Abadi, Computer Science Department, University of California at Santa Cruz, Santa Cruz, CA 95064, USA. Bogdan Warinschi, Department of Computer Science, University of Bristol, Merchant Venturers Building, Woodland Road, Bristol, BS8 1UB, UK.
A preliminary version of this paper appears in the Proceedings of the 24th ACM Symposium on Principles of Database Systems.

tized versions of the data would be produced, each corresponding to a partial view suitable for distribution to a subset of the clients. This naive scheme is impractical in general. Accordingly, there has been much interest in more elaborate and useful schemes for fine-grained control on access to published documents, particularly for XML documents [Bertino et al. 2002; Bertino et al. 2001; Crampton 2004; Damiani et al. 2002; Kudo and Hada 2000; Miklau and Suciu 2003; Yang and Li 2004]. This line of research has led to efficient and elegant publication techniques that avoid data duplication by relying on cryptography. For instance, using those techniques, medical records may be published as XML documents, with parts encrypted in such a way that only the appropriate users (physicians, nurses, researchers, administrators, and patients) can see their contents.

The work of Miklau and Suciu [2003] is a crisp, compelling example of this line of research. They develop a policy query language for specifying fine-grained access policies on XML documents and a logical model based on the concept of "protection". They also show how to translate consistent policies into protections, and how to implement protections by XML encryption [Eastlake and Reagle 2002]. Roughly, a protection is an XML tree in which nodes are guarded by positive boolean formulas over a set of symbols $\{K_1, K_2, \ldots\}$ that stand for cryptographic keys. Protections have a simple and clear intended semantics: access to the information contained in a node is conditioned on possession of a combination of keys that satisfies the formula that guards the node. For example, access to a node guarded by $(K_1 \wedge K_2) \vee K_3$ requires possessing either keys K_1 and K_2 or key K_3. (See Gifford's work for some of the roots of this approach [Gifford 1982].) Formally, a protection describes a function that maps each possible set of keys to the set of nodes that can be accessed using those keys, treating the keys as symbols. On the other hand, the use of keys for deriving a partially encrypted document is not symbolic: this process includes replacing the symbols $K_1, K_2, \ldots$ with actual keys, and applying a symmetric encryption algorithm repeatedly, bottom-up, to the XML document in question.

While Miklau and Suciu provide a thorough analysis of the translation of policies into protections, they leave a large gap between the abstract semantics of protections and the use of actual keys and encryption. The existence of this gap should not surprise us: an analogous gap existed in protocol analysis for 20 years, until recent efforts to bridge it (e.g., [Abadi and Rogaway 2002; Backes et al. 2003; Herzog 2004; Laud 2004; Micciancio and Warinschi 2004]). Concretely, the gap means that the protection semantics leaves many problematic issues unresolved. We describe two such issues, as examples:

—Partial information: It is conceivable that even when a node should be hidden according to a protection, the partially encrypted document may in fact leak some information about the data in that node.

—Encryption cycles: From the point of view of the abstract semantics, encryption cycles (such as encrypting a key with itself) are legitimate and do not contradict security. On the other hand, there are encryption algorithms that satisfy standard cryptographic definitions of security but that leak keys when encryption cycles are created (see Section 4).

More generally, there are many encryption methods and many notions of security

for them (e.g., [Bellare and Rogaway 2005; Dolev et al. 2000; Goldwasser and Micali 1984]), and it is not clear which one, if any, provides adequate guarantees for this application—nor is it exactly clear what those guarantees might be.

The immediate goal of this work is to bridge this gap by reconciling the abstract semantics of protections with a more concrete, computational treatment of security, and to define and establish precise security guarantees. We do not wish to replace the abstract semantics, which certainly has its place, but rather to complement it.

From a broader perspective, our goal is to develop, apply, and promote useful concepts and tools for security analysis in the field of database theory. These concepts and tools do not pertain to statistical techniques, which have long been known in database research (e.g., [Adam and Worthmann 1989; Castano et al. 1995; Ullman 1983]), but rather to cryptography. While sophisticated uses of cryptology in database research may have been of modest scope, there is an obvious need for database security, and we believe that cryptology has much to offer. In research on cryptographic protocols, formal and complexity-theoretic methods have been successful in providing detailed models and in enabling security proofs (sometimes automated ones). The same methods are beneficial for a broad class of systems that require security (e.g., [Micciancio and Panjwani 2006]). Each application, however, can necessitate non-trivial, specific insights and results. In the techniques that we study, partial and multiple encryptions occur in (large, XML) data instances; we therefore depart from the situations most typically considered in the cryptography literature, towards data management. It is this specificity that motivates the present paper.

Overview of Results

Our analysis is directed at the core of the framework of Miklau and Suciu, which aims to ensure data protection by an interesting combination of encryption schemes and secret sharing schemes [Shamir 1979].

As a formal counterpart to their loose, informal concept of data secrecy, we introduce a strong, precise cryptographic definition. The definition goes roughly as follows. Consider a protection for an XML document. An adversary is given an arbitrary set of keys, and the liberty of selecting two instantiations for the data in all nodes that occur in the XML document. The only restriction on these instantiations is that they should coincide on the nodes to which the adversary rightfully has access according to its keys and the abstract semantics of protections. In other words, the adversary selects two documents that contain the same information in the nodes it can access but may differ elsewhere. Then the adversary is given the partially encrypted document that corresponds to one of its two documents, and its goal is to decide which of the two instantiations was used in generating this partially encrypted document. Security means that the adversary cannot do much better than picking at random. It implies that the partially encrypted document reveals no information on the data in the nodes that should be hidden from the adversary, for otherwise this information would be sufficient to determine which instantiation was used.

Technically, we adapt and extend the approach of Abadi and Rogaway [2002], which has been followed and developed in several pieces of recent work (e.g., [Laud 2004; Micciancio and Panjwani 2005; 2006]). The novelties of this paper include the

application to document access control, significant differences in basic definitions motivated by this application, and the treatment of secret sharing. First we provide an intermediate symbolic language for cryptographic expressions. We then define patterns of expressions; intuitively, a pattern represents the information that an expression reveals to an adversary. We show how to transform protections into cryptographic expressions, and use patterns for providing an equivalent semantics for protections. This equivalence is captured in Theorem 1. Going further, we relate expressions to concrete computations on bit-strings. The most difficult result of this paper is Theorem 2. Informally, it states that patterns faithfully represent the information that expressions reveal, even when expressions and patterns are implemented with actual encryption schemes (not symbolically). More precisely, we associate probability distributions with an expression and its pattern by mapping symbols to bit-strings and implementing encryption with a semantically secure encryption scheme [Goldwasser and Micali 1984], and prove that these distributions cannot be distinguished by any probabilistic polynomial-time algorithm. Our main theorem, Theorem 4, reconciles the abstract semantics of protections with the actual use of encryption. We establish that if data is hidden according to a protection, then it is secret according to our definition of secrecy.

Contents

The next section, Section 2, is mostly a review. Section 3 introduces our formal language for representing cryptographic expressions and defines an alternative semantics of XML protections. Our main results are in Section 4, which gives concrete interpretations to expressions and relates the formal semantics of protections to a strong definition of secrecy. Section 5 considers some variants and extensions. Section 6 concludes. The Appendix contains additional proofs.

2. CONTROLLING ACCESS TO XML DOCUMENTS WITH PROTECTIONS

In this section we briefly recall the key aspects of the work of Miklau and Suciu. We focus on protections. We describe the derivation of partially encrypted documents from protections in the next section. We omit the policy query language because, for our purposes, we can discuss the protections generated from policies rather than the policies themselves.

XML Documents and Protections

We model XML documents as trees labeled with elements from a set $\mathsf{Data} = \{D_1, D_2, \ldots\}$, which we assume to represent XML element names and actual data values, though we make no syntactic distinction between these two possibilities. We treat D_1, D_2, ... as atomic data symbols. We use pre-order representations for XML trees, described by the grammar:

$$\mathsf{XML} ::= (\mathsf{Data}) \mid (\mathsf{Data}, \mathsf{XML}, \mathsf{XML}, \ldots, \mathsf{XML})$$

with terminals in $\mathsf{Data} \cup \{\text{“(”}, \text{“)”}\}$.

A *protection* consists of two components: a metadata XML tree obtained from an XML tree by adding metadata nodes, and a mapping that attaches to each node a positive boolean formula over $\mathsf{Keys} = \{K_1, K_2, \ldots\}$. Metadata nodes are nodes that have special kinds of labels, and it is assumed that they can be distinguished

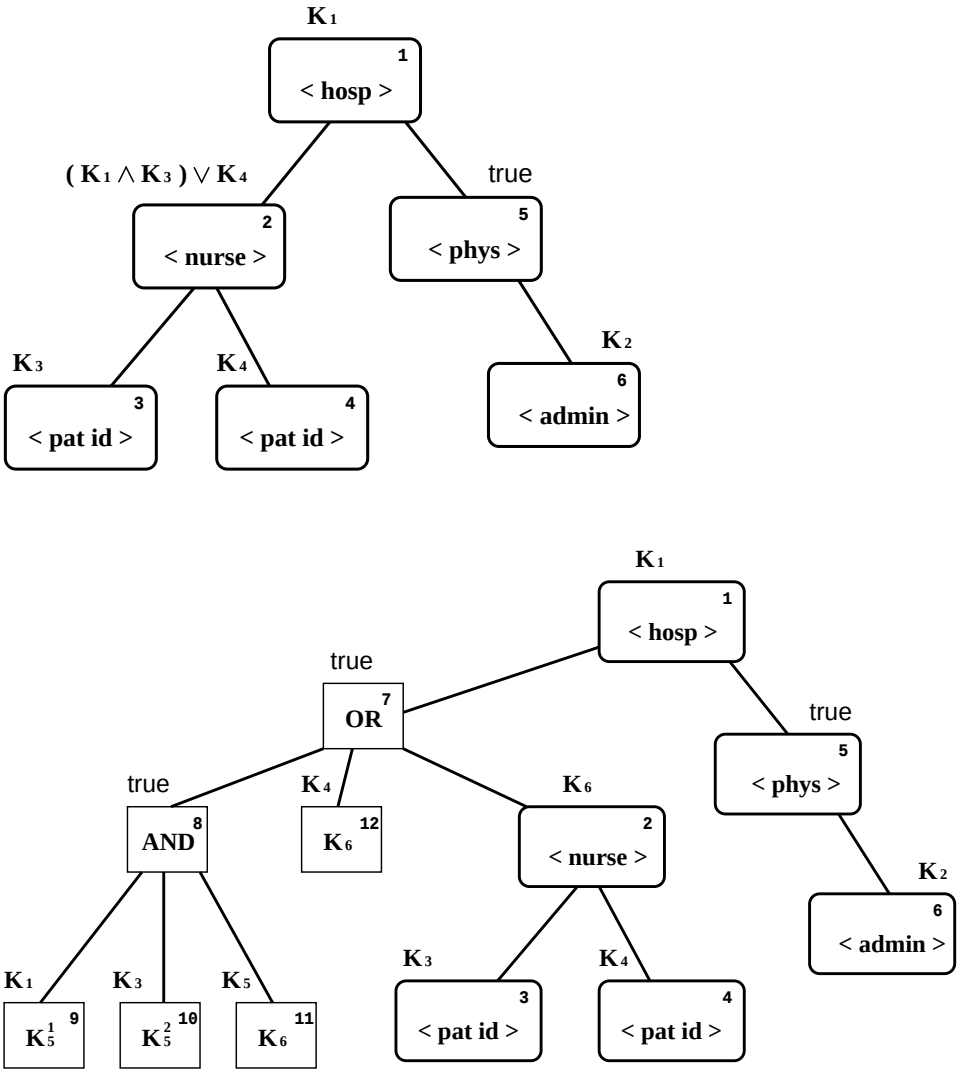

Fig. 1. A tree protection (top) and an equivalent normalized one (bottom).

$$\{D_1,\\ (\mathsf{OR},(\mathsf{AND},(\{K_5^1\}_{K_1},\{K_5^2\}_{K_1},\{K_6\}_{K_5}),\{K_6\}_{K_4},\{D_2,(\{D_3\}_{K_3},\{D_4\}_{K_4})\}_{K_6}),\\ (D_5,(\{D_6\}_{K_2})))\}_{K_1}$$

Fig. 2. Expression associated with the normalized protection of Figure 1.

from the standard XML nodes. Their labels are one of the symbols OR and AND, or hold keys in the set Keys or key shares in the set KeyShares defined by:

$$\mathsf{KeyShares} = \{K_i^j \mid K_i \in \mathsf{Keys}, j \in 1..n\}$$

for a fixed parameter n. Thus, in summary, protections are generated by the grammar:

$$\begin{aligned} \mathsf{Prot} &::= [(\mathsf{BData}), \mathsf{Cond}] \mid \\ &\quad\ [(\mathsf{BData}, \mathsf{Prot}, \mathsf{Prot}, \ldots, \mathsf{Prot}), \mathsf{Cond}] \\ \mathsf{Cond} &::= \mathsf{true} \mid \mathsf{false} \mid \mathsf{Keys} \mid \mathsf{Cond} \wedge \mathsf{Cond} \mid \mathsf{Cond} \vee \mathsf{Cond} \end{aligned}$$

where

$$\mathsf{BData} = \mathsf{Data} \cup \mathsf{Keys} \cup \mathsf{KeyShares} \cup \{\mathsf{OR}, \mathsf{AND}\}$$

is the set of node labels.

Roughly, the key shares K_i^1, K_i^2, ..., K_i^n are pieces of information that together allow the recovery of the key K_i, but of which no proper subset suffices for computing K_i, or even non-trivial partial information about K_i. For example, the key shares K_i^1, ..., K_i^{n-1} may be random strings of the same length as K_i, and K_i^n may be the XOR of K_i^1, ..., K_i^{n-1} with K_i. We treat a framework more general than the original one, which uses a particular way of sharing keys. We assume that the number of shares for each key is some integer constant n (with $n \geq 2$), and that each key is shared only once. (This assumption holds in the original framework, where $n = 2$.)

In a protection, the formula that guards a node describes a condition that needs to be satisfied before a user is granted access to that node (and its children). For example, accessing a node guarded by the formula $K_1 \wedge (K_2 \vee K_3)$ requires having key K_1 and at least one of the keys K_2 and K_3. In addition, the user should also satisfy all formulas on the path from the root to the node.

Figure 1 gives an example (adapted from [Miklau and Suciu 2003]). The tree at the top is an example of a protection over some XML medical database. Notice that we preserved the original labels—we did not replace element names with symbols D_1, D_2, ... A user that possesses only key K_3 cannot access any node since the root is guarded by K_1. If the user knows also K_1 then it should be able to access the data in nodes $\{1, 2, 3, 5\}$.

Normalization

Using simple transformations, one can rewrite any protection into an equivalent, *normalized* protection where all formulas that guard nodes are atomic, that is, one of true, false, or K for some $K \in \mathsf{Keys}$. Normalization requires adding metadata nodes, keys, and key shares. Normalization can also include removing parts guarded by false, so we assume that false does not appear in normalized formulas (departing slightly from the original definition but without loss of generality). Normalized protections are important because, in standard encryption schemes, one can encrypt under an atomic key but not under a boolean combination of keys. Normalized protections serve as the basis for producing partially encrypted documents by applying an encryption algorithm repeatedly.

In Figure 1, the tree at the bottom is a normalized protection, equivalent to the protection at the top. In the normalized protection, a user with keys K_1 and K_3 can recover the information in nodes 9 and 10, that is, the key shares K_5^1 and K_5^2, and therefore K_5. (Recall that, in [Miklau and Suciu 2003], keys are split into only two shares.) The key K_5 can then be used to recover the information in node 11, that is, the key K_6 which together with K_1 provides access to the nodes $\{1, 2, 3, 5\}$ of the original tree.

A Semantics of Protections

Formally, the semantics of a protection P is the function $\mathsf{Acc}_P : \mathcal{P}(\mathsf{Keys}) \to \mathcal{P}(\mathsf{Data})$ that, given a set of keys $T \subseteq \mathsf{Keys}$, returns the set of data that can be accessed using the keys in T. Computing the function $\mathsf{Acc}_P(T)$ is an iterative process. The keys in T are used to access new keys (by either obtaining them directly or recovering all their shares), and the process is repeated until a maximal set of keys is obtained. The output of $\mathsf{Acc}_P(T)$ is the set of all data contained in nodes that can be accessed using this last set of keys.

We refer the reader to Miklau and Suciu [2003] for additional details on the definition of $\mathsf{Acc}_P(T)$. In most of our analysis, we use an equivalent formulation of the formal semantics, given in Section 3.

On Keys Derived from Data

In the description above and in the analysis that follows, we do not consider the use of keys derived from data (for example, from mother's maiden names and social security numbers). There are at least three obstacles to obtaining security guarantees with such keys. These obstacles are not specific to a particular scheme, but they do arise in this context.

—First, the potential lack of entropy in data implies that the resulting keys may offer no security. For example, keys computed from 9-digit social security numbers can have at most 10^9 values, which an attacker may try in order to recover anything encrypted under those keys. An attacker may even learn the underlying social security number when decryption with one of those values succeeds.

—Moreover, the key derivations can lose some data entropy. Specifically, Miklau and Suciu compute a 128-bit key by breaking data into 128-bit blocks and XORing those blocks; if a piece of data yields blocks in which only the first m bits are meaningful and the last $128 - m$ bits contain a fixed pattern, then the resulting key has at most m bits of entropy, no matter how much entropy was initially present in the data.

—Finally, the resulting keys can be involved in questionable encryption cycles—for instance, protecting a mother's maiden name with a social security number and vice versa.

3. FORMAL ANALYSIS

In this section we introduce a language for representing cryptographic expressions that use symmetric encryption and secret sharing schemes. We rely on this language for providing a formal account of the transformation of protections into partially encrypted documents. We also define patterns of expressions, which capture the

information that expressions reveal to an adversary, in symbolic terms. This definition is a variant of ones suggested in other contexts [Abadi and Rogaway 2002; Herzog 2004; Micciancio and Panjwani 2005]. Finally, we establish a relation between the semantics of protections and these patterns.

Basically, expressions are useful as an intermediate language between protections and the world of probability distributions on bit-strings. They also serve as a bridge to other work on formal analysis. That work includes the idea of patterns, as generalizations of expressions. We adapt and extend that idea for the present setting.

An Intermediate Cryptographic Language

We consider expressions built from the set of basic data BData (defined above), using tupling and encryption with keys in Keys. The grammar for expressions is:

$$\mathsf{Exp} ::= \mathsf{BData} \mid (\mathsf{Exp}, \mathsf{Exp}, \ldots, \mathsf{Exp}) \mid \{\mathsf{Exp}\}_{\mathsf{Keys}}$$

For example, an element in Exp is the expression

$$(\{(D_2, K_1^2)\}_{K_2}, \{(D_1, \{K_2\}_{K_3})\}_{K_1})$$

It represents the tupling of two ciphertexts. The first ciphertext is the encryption under key K_2 of data D_2 and key share K_1^2. The second ciphertext is the encryption under key K_1 of the tuple formed from data D_1 and the encryption under key K_3 of key K_2.

We sometimes omit parentheses in expressions. For example, we may write $\{D_1, K_1\}_{K_2}$ rather than $\{(D_1, K_1)\}_{K_2}$.

Associating Cryptographic Expressions to Protections

We use our language of expressions for giving a precise definition of how to map a normalized protection to (a symbolic representation for) a partially encrypted document. This definition is recursive. It relies on the intuition that each node of the protection under consideration is (recursively) mapped to a part of the final, partially encrypted document. If the node is guarded by a key, then the corresponding part of the final document is encrypted under that key, otherwise it is left in clear.

Formally, we define the function $\mathsf{E} : \mathsf{Prot} \to \mathsf{Exp}$ inductively by:

—$\mathsf{E}([(B, P_1, P_2, \ldots, P_l), \mathsf{true}]) = (B, \mathsf{E}(P_1), \mathsf{E}(P_2), \ldots, \mathsf{E}(P_l))$
—$\mathsf{E}([(B, P_1, P_2, \ldots, P_l), K]) = \{(B, \mathsf{E}(P_1), \mathsf{E}(P_2), \ldots, \mathsf{E}(P_l))\}_K$

where $l \geq 0$, each P_i (for $i = 1, 2, \ldots, l$) is a normalized protection, B is an arbitrary symbol in BData, and K is an arbitrary key in Keys.

For example, the expression of Figure 2 corresponds to the second protection in Figure 1, with the element name contained in node i replaced with data symbol D_i.

Cycles

The definition of expressions allows encryption cycles of the kind discussed in the introduction. For some of our results, it is useful to focus on a class of acyclic expressions:

Definition 1 Acyclic expressions. *A plain occurrence of a key K_j in an expression E_1 is one where K_j is not the subscript in a subexpression $\{\ldots\}_{K_j}$. Key*

K_i encrypts key K_j in expression E if there exists a subexpression $\{E_1\}_{K_i}$ in E such that K_j occurs plainly in E_1 or K_j^k (for some $k \in 1..n$) occurs in E_1. The expression E is acyclic if the graph associated with its "encrypts" relation is acyclic.

For example, $\{K_1\}_{K_2}$ and $\{\{K_1\}_{K_2}\}_{K_2}$ are both acyclic expressions, while $\{K_1\}_{K_1}$ and $(\{K_1^1\}_{K_2}, \{K_2\}_{K_1})$ are not.

As long as data values are not used as keys, a protection P that results from the translation of a policy never contains keys in its nodes. In turn, if normalizing P yields P', then $\mathsf{E}(P')$ is an acyclic expression. (This point follows from the definition of normalization [Miklau and Suciu 2003].)

Recoverable Keys

Given an expression E, we write $\mathsf{keys}(E)$ for the set of key symbols that occur in E and key symbols whose shares occur in E. We call a key *recoverable* if the key occurs in clear (that is, not encrypted), if the key occurs encrypted under recoverable keys, or if each of its shares occurs in clear or encrypted under recoverable keys. More precisely, a key is recoverable if it is in the least set S of expressions such that:

—$E \in S$,
—for all $E_1, \ldots, E_m \in \mathsf{Exp}$, if $(E_1, \ldots, E_m) \in S$, then $E_1 \in S$, ..., $E_m \in S$,
—for all $E' \in \mathsf{Exp}$ and $K_j \in \mathsf{Keys}$, if $\{E'\}_{K_j} \in S$ and $K_j \in S$, then $E' \in S$,
—for all $K_j \in \mathsf{Keys}$, if $K_j^1, \ldots, K_j^n \in S$, then $K_j \in S$.

We write $\mathsf{recoverable}(E)$ for the set of all recoverable keys in expression E.

For example, with $n = 2$, for the expression

$$E = (\{K_1, K_2^1, K_6^1, K_4\}_{K_3}, \{K_2^2\}_{K_2}, K_3, \{K_5\}_{K_4})$$

we have

$$\mathsf{keys}(E) = \{K_1, K_2, K_3, K_4, K_5, K_6\}$$

and

$$\mathsf{recoverable}(E) = \{K_1, K_3, K_4, K_5\}$$

while for the expression

$$E' = (\{K_1, K_2^1, K_6^1, K_4\}_{K_3}, \{K_5\}_{K_2}, K_3, \{K_2^2\}_{K_4})$$

we have

$$\mathsf{keys}(E') = \{K_1, K_2, K_3, K_4, K_5, K_6\}$$

and

$$\mathsf{recoverable}(E') = \{K_1, K_2, K_3, K_4, K_5\}$$

Expression Patterns

For each expression E, we define its *structure* $\mathsf{struct}(E)$. Essentially, the structure of an expression is given by its parse tree in which the labels are replaced with fresh symbols. We therefore introduce D, K_0, and K_0^j (for $j \in 1..n$), all disjoint from BData, and define the structure of expressions by:

—$\mathsf{struct}(K_i) = K_0$ for all $K_i \in \mathsf{Keys}$,
—$\mathsf{struct}(K_i^j) = K_0^j$ for all $K_i \in \mathsf{Keys}$, $j \in 1..n$,
—$\mathsf{struct}(D_i) = D$ for all $D_i \in \mathsf{Data}$,
—$\mathsf{struct}(\mathsf{OR}) = \mathsf{OR}$,
—$\mathsf{struct}(\mathsf{AND}) = \mathsf{AND}$,
—$\mathsf{struct}((E_1, E_2, \ldots, E_m)) = (\mathsf{struct}(E_1), \mathsf{struct}(E_2), \ldots, \mathsf{struct}(E_m))$,
—$\mathsf{struct}(\{E\}_{K_i}) = \{\mathsf{struct}(E)\}_{K_0}$ for all $K_i \in \mathsf{Keys}$.

Thus, data symbols are replaced with D, key symbols are replaced with K_0, and key shares are replaced with corresponding shares of K_0. For example, the structure of the expression

$$\{(\{K_1, D_2\}_{K_3}, K_3^1)\}_{K_2}$$

is

$$\{(\{K_0, D\}_{K_0}, K_0^1)\}_{K_0}$$

The encryption cycle in this structure is unimportant for our purposes. Alternative definitions of expression structure can avoid such cycles without affecting our results. (For example, one such definition lets the structure of $\{E\}_{K_i}$ be $\{\mathsf{struct}(E)\}_{K_0'}$, where K_0' is a fresh symbol distinct from K_0.)

We write $\mathsf{p}(E, T)$ for the *pattern* that can be observed in expression E using for decryption the keys in $T \subseteq \mathsf{keys}(E)$. The set of patterns is defined like the set of expressions but over extended sets of atomic symbols that include D, K_0, and K_0^j (for $j \in 1..n$). The pattern $\mathsf{p}(E, T)$ is defined by:

—$\mathsf{p}(B, T) = B$ for all $B \in \mathsf{BData}$,
—$\mathsf{p}((E_1, E_2, \ldots, E_m), T) = (\mathsf{p}(E_1, T), \mathsf{p}(E_2, T), \ldots, \mathsf{p}(E_m, T))$,
—$\mathsf{p}(\{E\}_{K_i}, T) = \{\mathsf{struct}(E)\}_{K_i}$ if $K_i \notin T$,
—$\mathsf{p}(\{E\}_{K_i}, T) = \{\mathsf{p}(E, T)\}_{K_i}$ if $K_i \in T$.

The idea that motivates this definition is that a ciphertext encrypted under an unknown key K_i reveals at most the structure of the underlying plaintext and the identity of K_i, but not the encrypted value. Section 4 explains a computational counterpart of this idea.

We write $\mathsf{pattern}(E)$ for the pattern obtained from E by using for decryption the keys recoverable from E itself. We let:

$$\mathsf{pattern}(E) = \mathsf{p}(E, \mathsf{recoverable}(E))$$

For example, the pattern of the expression

$$(\{D_1\}_{K_1}, \{D_2\}_{K_2}, K_1)$$

is

$$(\{D_1\}_{K_1}, \{D\}_{K_2}, K_1)$$

Similarly, the pattern of the expression

$$(\{\{D_1\}_{K_2}\}_{K_1}, \{D_2\}_{K_2}, K_1)$$

is

$$(\{\{D\}_{K_2}\}_{K_1}, \{D\}_{K_2}, K_1)$$

Finally, when $n = 2$, the pattern of the expression

$$(\{D_1\}_{K_1}, \{\{K_1\}_{K_2}\}_{K_3}, K_3^1, K_3^2, \{(\{K_1, D_2\}_{K_3}, K_3^1)\}_{K_2})$$

is

$$(\{D\}_{K_1}, \{\{K_0\}_{K_2}\}_{K_3}, K_3^1, K_3^2, \{(\{K_0, D\}_{K_0}, K_0^1)\}_{K_2})$$

Secrecy, Symbolically

Using patterns, we characterize the set of visible data (and thus the set of secret data) in formal expressions. By analogy with the definition of $\mathsf{Acc}_P(T)$, we define the set of data accessible in expression E with keys in the set $\{K_1, K_2, \ldots, K_l\} \subseteq \mathsf{keys}(E)$ by:

$$\mathsf{Acc}_E(\{K_1, K_2, \ldots, K_l\}) = \{D_i \in \mathsf{Data} \mid D_i \text{ occurs in } \mathsf{pattern}(E, K_1, K_2, \ldots, K_l)\}$$

The following theorem relates $\mathsf{Acc}_P(T)$ and $\mathsf{Acc}_E(T)$, thus justifying the similarity in notation. It states that the set of data that can be accessed with keys K_1, K_2,..., K_l according to protection P is precisely the set of data that can be seen in the pattern of $(\mathsf{E}(P), K_1, K_2, \ldots, K_l)$. The purpose of the tupling with the keys K_1, K_2,..., K_l is to make those keys immediately recoverable.

THEOREM 1. *Let P be a normalized protection and $T \subseteq \mathsf{keys}(\mathsf{E}(P))$. Then $\mathsf{Acc}_P(T) = \mathsf{Acc}_{\mathsf{E}(P)}(T)$.*

When T is empty, the theorem holds because $\mathsf{Acc}_P(\emptyset)$ and $\mathsf{Acc}_{\mathsf{E}(P)}(\emptyset)$ both consist of the data that occurs only under keys that can be recovered from $\mathsf{E}(P)$. More generally, for an arbitrary T, it holds because $\mathsf{Acc}_P(T)$ and $\mathsf{Acc}_{\mathsf{E}(P)}(T)$ both consist of the data that occurs only under keys that can be recovered from the combination of $\mathsf{E}(P)$ and T. We omit a complete proof of this theorem; in the rest of the paper we rely primarily on $\mathsf{Acc}_E(T)$ rather than on $\mathsf{Acc}_P(T)$.

4. COMPUTATIONAL ANALYSIS

In this section we give a concrete interpretation for expressions and patterns as probability distributions on bit-strings. This interpretation is computational in the sense that it relies on computations on bit-strings rather than on symbolic expressions. In particular, encryption is an actual computation on bit-strings, rather than a formal operation. In this computational world, we give a cryptographic definition for data secrecy, then prove our main results.

We first recall the definition of computational indistinguishability (a central ingredient for defining cryptographic secrecy) and those of encryption and secret sharing schemes.

Indistinguishability of Distribution Ensembles

Let $\{\mathcal{D}_\eta^0\}_\eta$ and $\{\mathcal{D}_\eta^1\}_\eta$ be two distribution ensembles on bit-strings (sequences of probability distributions on bit-strings, indexed by a security parameter η). We

write $x \stackrel{R}{\leftarrow} \mathcal{D}^i_\eta$ to indicate that x is sampled according to $\mathcal{D}^i_\eta$. Let A be an algorithm. The advantage of A in distinguishing between the two ensembles is the quantity:

$$\mathbf{Adv}^{\mathrm{dist}}_{\mathcal{D}^0,\mathcal{D}^1}(A,\eta) = \Pr\left[x \stackrel{R}{\leftarrow} \mathcal{D}^0_\eta : A(x,\eta) = 1\right] - \Pr\left[x \stackrel{R}{\leftarrow} \mathcal{D}^1_\eta : A(x,\eta) = 1\right]$$

We say that $\mathcal{D}^0$ and $\mathcal{D}^1$ are computationally indistinguishable, and write $\mathcal{D}^0 \approx \mathcal{D}^1$, if for any probabilistic polynomial-time algorithm A the quantity $\mathbf{Adv}^{\mathrm{dist}}_{\mathcal{D}^0,\mathcal{D}^1}(A,\eta)$ is negligible as a function of η. (A function $f(\eta)$ is negligible if it is smaller than the inverse of any polynomial for all sufficiently large inputs η [Bellare and Rogaway 2005].)

Encryption Schemes

An encryption scheme Π consists of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ for key generation, encryption, and decryption, respectively. The key generation algorithm is randomized; it takes as input a security parameter η and returns a key k to be used for both encryption and decryption. We write $k \stackrel{R}{\leftarrow} \mathcal{K}(\eta)$ for the process of generating encryption keys. The encryption algorithm is also randomized. It takes as input a key k and a plaintext m, and outputs a ciphertext c. We write $c \stackrel{R}{\leftarrow} \mathcal{E}(k,m)$ for the process of encrypting message m with key k, producing c. Finally, the decryption algorithm takes as input a key k and a ciphertext c. If $c \stackrel{R}{\leftarrow} \mathcal{E}(k,m)$ for a key k and a plaintext m, then $\mathcal{D}(k,c) = m$. Decryption returns $\perp$ if it does not succeed.

We use a standard notion of security for encryption: indistinguishability against chosen plaintext attacks (IND-CPA), also known as semantic security [Goldwasser and Micali 1984]. We define it next. For a given encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and a bit b, we consider a "left-right" oracle $LR_{\Pi,b}(\eta)$. This oracle is a program that generates a key k (via $k \stackrel{R}{\leftarrow} \mathcal{K}(\eta)$), and then answers queries of the form (m_0, m_1), where m_0 and m_1 are bit-strings of equal length. The oracle always returns the answer $\mathcal{E}(k, m_b)$, that is, the encryption of the message selected by b. Scheme Π is said to be IND-CPA secure if for any probabilistic polynomial-time adversary A with access to the oracle described above, the quantity:

$$\mathbf{Adv}^{\text{ind-cpa}}_{\Pi}(A,\eta) = \Pr\left[A^{LR_{\Pi,0}(\eta)}(\eta) = 1\right] - \Pr\left[A^{LR_{\Pi,1}(\eta)}(\eta) = 1\right]$$

is negligible as a function of η. The probabilities are induced by the randomness used in the adversary and in the key-generation and encryption processes. Intuitively, the adversary A does not know b a priori, and it aims to discover b. The advantage $\mathbf{Adv}^{\text{ind-cpa}}_{\Pi}(A,\eta)$ will not be negligible if A can determine b by interacting with the oracle. A fortiori, the advantage $\mathbf{Adv}^{\text{ind-cpa}}_{\Pi}(A,\eta)$ will not be negligible if A can break encryptions, and therefore determine b by decrypting the oracle's response to a query (m_0, m_1) where $m_0 \neq m_1$.

According to this definition of security, encryption need not hide the length of plaintexts (because m_0 and m_1 are required to be of equal length). Since length may depend on structure, the structure of plaintexts may be revealed as well. Furthermore, encryption need not hide the identity of keys: an adversary may be able to distinguish two encryptions under the same unknown key from two encryptions under different unknown keys. Finally, since the oracles are responsible for generating and using the encryption keys, the adversary cannot guess or invent a

key and submit it for encryption under itself; hence the definition offers no guarantee for such cyclic encryptions, which may therefore leak keys. Alternative notions of security can yield stronger guarantees, but correspondingly they are harder to satisfy. Section 5 considers some such notions.

Secret Sharing Schemes

An n-out-of-n secret sharing scheme $\mathcal{SS} = (\mathcal{S}, \mathcal{C})$ for sharing keys of Π consists of algorithms for share creation and share combination. The randomized share creation algorithm $\mathcal{S}$ takes as input a key k and the security parameter η used to generate it, and outputs n shares of k: k^1, k^2, ..., k^n. The share combination algorithm $\mathcal{C}$ takes as input n shares $k^1, k^2, \ldots, k^n$, and attempts to reconstitute the original key. The scheme is correct if $\mathcal{C}(\mathcal{S}(k, \eta)) = k$ for any key k and any η.

Security of secret sharing means that a proper subset of the shares for a key provides no useful information on the key—more precisely, we require that this subset cannot be distinguished from a corresponding subset for another key. Let $sh(k)$ be a sample from the distribution $\mathcal{S}(k, \eta)$ of the n shares output by the sharing algorithm for the key k, and $sh(k)|_S$ be the restriction of $sh(k)$ to the indexes in some set of indexes $S \subseteq \{1, 2, \ldots, n\}$. We require that, for any proper subset of indexes $S \subset \{1, 2, \ldots, n\}$ and any probabilistic polynomial-time adversary A, the quantity:

$$\begin{aligned}\mathbf{Adv}^{\mathrm{ss}}_{\mathcal{SS}}(A, \eta) \;=\; &\Pr\left[k_0, k_1 \xleftarrow{R} \mathcal{K}(\eta), sh(k_0) \xleftarrow{R} \mathcal{S}(k_0, \eta) : A(k_0, k_1, sh(k_0)|_S) = 1)\right] - \\ &\Pr\left[k_0, k_1 \xleftarrow{R} \mathcal{K}(\eta), sh(k_1) \xleftarrow{R} \mathcal{S}(k_1, \eta) : A(k_0, k_1, sh(k_1)|_S) = 1)\right]\end{aligned}$$

is negligible as a function of η.

While this security requirement does not seem to be well-established, it should not be too surprising. Particular schemes may have additional properties, for example that $sh(k_0)|_S$ and $sh(k_0)|_{S'}$ are indistinguishable whenever S and S' are proper subsets of indexes of the same size (so key shares may all look alike). We do not need those properties.

Computational Interpretation of Expressions

Expressions and patterns induce distributions on bit-strings. These distributions are obtained by replacing data symbols with bit-strings and implementing encryption and key sharing with actual encryption and secret sharing schemes.

Formally, for any expression or pattern E, given a function $f : \mathsf{Data} \to \{0,1\}^*$ that maps data symbols to bit-string representations of XML element names and values, an encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, a secret sharing scheme $\mathcal{SS} = (\mathcal{S}, \mathcal{C})$, and a security parameter η, we define the distribution $[\![E]\!]_f^{\Pi, \mathcal{SS}, \eta}$ (and thus a distribution ensemble $[\![E]\!]_f^{\Pi, \mathcal{SS}}$) using a two-step procedure:

(1) In the first step, each key symbol is mapped to a bit-string. Specifically, we assume that $\mathsf{keys}(E) = \{K_1, \ldots, K_m\}$, and we generate a vector τ from the distribution $\mathcal{K}^{m+1}(\eta)$. This is a vector of $m+1$ keys, each obtained by running the key generation algorithm on the security parameter. We map K_i to $\tau[i]$, and in particular map K_0 to $\tau[0]$. We then obtain shares of the keys in τ

$$
\begin{aligned}
[\![D]\!]_f^{\Pi,\mathcal{SS},\eta} &= \texttt{data} \\
[\![D_i]\!]_f^{\Pi,\mathcal{SS},\eta} &= f(D_i) \\
[\![\mathsf{OR}]\!]_f^{\Pi,\mathcal{SS},\eta} &= \texttt{or} \\
[\![\mathsf{AND}]\!]_f^{\Pi,\mathcal{SS},\eta} &= \texttt{and} \\
[\![K_i]\!]_f^{\Pi,\mathcal{SS},\eta} &= \tau[i] \\
[\![K_i^j]\!]_f^{\Pi,\mathcal{SS},\eta} &= \phi[i][j] \\
[\![(E_1, E_2, \ldots, E_l)]\!]_f^{\Pi,\mathcal{SS},\eta} &= ([\![E_1]\!]_f^{\Pi,\mathcal{SS},\eta}, [\![E_2]\!]_f^{\Pi,\mathcal{SS},\eta}, \ldots, [\![E_l]\!]_f^{\Pi,\mathcal{SS},\eta}) \\
[\![\{E\}_{K_i}]\!]_f^{\Pi,\mathcal{SS},\eta} &= \mathcal{E}(\tau[i], [\![E]\!]_f^{\Pi,\mathcal{SS},\eta})
\end{aligned}
$$

Fig. 3. Mapping expressions to bit-strings.

by running the secret sharing scheme $\mathcal{SS}$; these shares are maintained in an $(m+1)$-by-n matrix ϕ whose rows are obtained by $\phi[i] \stackrel{R}{\leftarrow} \mathcal{S}(\tau[i], \eta)$.

(2) In the second step, we map each expression (or pattern) E to an interpretation $[\![E]\!]_f^{\Pi,\mathcal{SS},\eta}$ that we define inductively in Figure 3. This step assumes constant bit-strings "`or`", "`and`", and "`data`", as well as a tupling operation on bit-strings.

The computational interpretation $[\![\mathsf{E}(P)]\!]_f^{\Pi,\mathcal{SS},\eta}$ associated with the expression $\mathsf{E}(P)$ is the distribution of the partially encrypted document derived from protection P, generated with encryption scheme Π, secret sharing scheme $\mathcal{SS}$, and η as security parameter.

Because encryption may not hide the length of plaintexts, we typically need hypotheses on the lengths of bit-string representations. For simplicity, we assume that those lengths depend only on structure. More precisely, we assume that the length of $[\![E]\!]_f^{\Pi,\mathcal{SS},\eta}$ always equals the length of $[\![\mathsf{struct}(E)]\!]_f^{\Pi,\mathcal{SS},\eta}$.

In the case where E is a data symbol, this assumption implies that we focus on functions $f : \mathsf{Data} \to \{0,1\}^*$ that map data symbols to bit-strings of a fixed length:

Definition 2. *A valuation is a function $f : \mathsf{Data} \to \{0,1\}^*$ that maps every data symbol to a bit-string of the same length as* `data`.

In the case where E is a key symbol, the assumption means that the key generator yields keys of a fixed length for each given security parameter. A similar condition applies to key shares. In the case where E is a tuple, it suffices that the length of a tuple be a function of the lengths of its components. Similarly, in the case where E is an encryption, it suffices that the length of a ciphertext be a function of the length of the underlying plaintext and of the security parameter. These conditions on keys, key shares, tuples, and encryptions hold in most usual implementations.

Secrecy, Computationally

We use the concept of computational indistinguishability ($\approx$) and the computational interpretation of expressions for giving a computational characterization for secrecy

of data that occurs in expressions.

DEFINITION 3. *Let E be an expression, Π an encryption scheme, $\mathcal{SS}$ a secret sharing scheme, and $S \subseteq \mathsf{Data}$. The set S is computationally hidden in E (with Π and $\mathcal{SS}$) if for any two valuations f_0 and f_1 such that*

$$f_0(D_i) = f_1(D_i) \quad \textit{for all } D_i \in \mathsf{Data} - S$$

it holds that $[\![E]\!]_{f_0}^{\Pi,\mathcal{SS}} \approx [\![E]\!]_{f_1}^{\Pi,\mathcal{SS}}$.

As explained in the introduction, this definition indicates that an adversary is allowed to choose two interpretations for the data symbols in the expression E. These interpretations must map data that is not secret to the same bit-strings, but may map other data to different bit-strings of the same length. The adversary is then given a bit-string selected from the distribution determined by one of the two interpretations and its goal is to determine which interpretation was used. Secrecy means that the adversary cannot do much better than guessing.

Main Results

The technical core of our results is the next theorem. It states that patterns faithfully represent the information that expressions reveal, even when expressions and patterns are mapped to bit-strings. Specifically, we prove that the distribution ensembles associated with E and $\mathsf{pattern}(E)$ are indistinguishable.

THEOREM 2. *Let E be an acyclic expression. If Π is an* IND-CPA *secure encryption scheme and $\mathcal{SS}$ is a secure secret sharing scheme, then for any valuation f it holds that*

$$[\![E]\!]_{f}^{\Pi,\mathcal{SS}} \approx [\![\mathsf{pattern}(E)]\!]_{f}^{\Pi,\mathcal{SS}}$$

The proof of this theorem relies on a so-called hybrid argument. It is presented in the Appendix.

Next, we build on Theorem 2 in order to establish a link between symbolic and computational notions of data secrecy.

LEMMA 3. *Let E be an acyclic expression and $T = \{K_1, K_2, \ldots, K_l\} \subseteq \mathsf{keys}(E)$ a set of keys. If Π is an* IND-CPA *secure encryption scheme and $\mathcal{SS}$ is a secure secret sharing scheme, then $\mathsf{Data} - \mathsf{Acc}_E(T)$ is computationally hidden in $(E, K_1, K_2, \ldots, K_l)$ with Π and $\mathcal{SS}$.*

This lemma follows from the definition of $\mathsf{Acc}_E(T)$ and Theorem 2. Since $\mathsf{Acc}_E(T)$ consists of the data symbols that occur in the pattern of $(E, K_1, K_2, \ldots, K_l)$, we have that:

$$[\![\mathsf{pattern}((E, K_1, K_2, \ldots, K_l))]\!]_{f_0}^{\Pi,\mathcal{SS}} = [\![\mathsf{pattern}((E, K_1, K_2, \ldots, K_l))]\!]_{f_1}^{\Pi,\mathcal{SS}}$$

for any two valuations f_0 and f_1 that coincide on $\mathsf{Acc}_E(T)$. We conclude by Theorem 2 and transitivity.

Going further, Theorem 4 relates the abstract semantics of a normalized protection P, as defined by the function $\mathsf{Acc}_P(\cdot)$, to the secrecy of data in the partially encrypted document associated with P. It requires that $\mathsf{E}(P)$ be acyclic, as we would expect for protections derived from policies (see Section 3). It states that if

some data is secret according to the abstract semantics of protections, then that data is in fact computationally hidden. Therefore, we regard Theorem 4 as the main theorem of this paper.

THEOREM 4. *Let P be a normalized protection such that $\mathsf{E}(P)$ is an acyclic expression. Let $T = \{K_1, K_2, \ldots, K_l\} \subseteq \mathsf{keys}(\mathsf{E}(P))$ be an arbitrary set of keys. If Π is an* IND-CPA *secure encryption scheme and $\mathcal{SS}$ is a secure secret sharing scheme, then $\mathsf{Data} - \mathsf{Acc}_P(T)$ is computationally hidden in $(\mathsf{E}(P), K_1, K_2, \ldots, K_l)$ with Π and $\mathcal{SS}$.*

This theorem is an immediate corollary of Theorem 1 and Lemma 3.

Consequences (Discussion)

It is important to understand what our results imply, and also what they do not imply. Next we discuss some of their consequences. We focus this discussion on the expression of Figure 2,

$$\{D_1, \\ (\mathsf{OR}, (\mathsf{AND}, (\{K_5^1\}_{K_1}, \{K_5^2\}_{K_1}, \{K_6\}_{K_5}), \{K_6\}_{K_4}, \{D_2, (\{D_3\}_{K_3}, \{D_4\}_{K_4})\}_{K_6}), \\ (D_5, (\{D_6\}_{K_2})))\}_{K_1}$$

which here we call E.

Theorem 4 implies that if $T = \{K_1, K_3\}$ and if f_0 and f_1 are two valuations that coincide on D_1, D_2, D_3, and D_5, then $[\![E]\!]_{f_0}^{\Pi,\mathcal{SS}} \approx [\![E]\!]_{f_1}^{\Pi,\mathcal{SS}}$. The two valuations may differ on D_4 and D_6. The theorem implies that no probabilistic polynomial-time adversary can get significant partial information on the values of D_4 and D_6 (beyond any information that it has a priori). In other words, the set $\{D_4, D_6\}$ is computationally hidden in E, as one might have expected.

The valuations f_0 and f_1 are allowed to result in some (partial or total) coincidences in values. For instance, we may have $f_0(D_4) = f_0(D_6)$ but $f_1(D_4) \neq f_1(D_6)$. The adversary cannot distinguish $[\![E]\!]_{f_0}^{\Pi,\mathcal{SS}}$ and $[\![E]\!]_{f_1}^{\Pi,\mathcal{SS}}$ despite these coincidences. Thus, value repetitions are concealed. In terms of the protection of Figure 1, the adversary cannot even tell whether the value of $\langle$`pat id`$\rangle$ of node 4 and the value of $\langle$`admin`$\rangle$ of node 6 are the same.

Since the valuations f_0 and f_1 are universally quantified, they may be chosen by the adversary. So, while the adversary obtains information about a document only by observing it, we allow for the possibility that the adversary may influence the contents of the document. This possibility seems realistic in many potential applications. For instance, in our particular example, the adversary may be the patient whose record is in node 4, and may therefore have some influence and prior knowledge on the contents of node 4.

On the other hand, the guarantees that the theorem offers are perhaps not as strong as some might expect, in particular with respect to document structure. For instance, the theorem does not exclude that an adversary might be able to distinguish a bit-string representation of E from a bit-string representation of the expression that we obtain from E by replacing D_6 with a non-atomic expression. Theorem 4 notwithstanding, an adversary may learn a great deal about the length and even the structure of encrypted material.

Unfortunately, this limitation of the theorem is not a shortcoming of our analysis, but rather the result of an intrinsic shortcoming in the technique that we are analyzing. Under the standard definition of security that we adopt for encryption, one cannot expect to do much better [Micciancio 2004]. As explained above, encryption need not hide the length of plaintexts. (Indeed, the XML encryption method used by Miklau and Suciu does not.) Moreover, if E_0 and E_1 are expressions with different structures, the concrete bit-string implementations $[\![E_0]\!]_f^{\Pi,\mathcal{SS},\eta}$ and $[\![E_1]\!]_f^{\Pi,\mathcal{SS},\eta}$ may well have different lengths, so it is possible that their encryptions could be distinguished. The next section discusses stronger—but less standard—assumptions on encryption, and also other variants to our definitions.

5. EXTENSIONS

Alternative definitions are certainly possible, and perhaps attractive. In this section we consider some alternative definitions and corresponding extensions of our main results. We study additional security properties, addressing more stringent secrecy requirements and some more powerful attacks.

Stronger Secrecy Requirements for Encryption

In particular, we may wish to require that a ciphertext encrypted under an unknown key reveal nothing about the underlying plaintext—not even its structure or its length [Abadi and Rogaway 2002].

Formally, we may replace the functions p and $\mathsf{pattern}$ of Section 3. For instance, we may define replacements p' and $\mathsf{pattern}'$ as follows:

—$\mathsf{p}'(B,T) = B$ for all $B \in \mathsf{BData}$,
—$\mathsf{p}'((E_1,E_2,\ldots,E_m),T) = (\mathsf{p}'(E_1,T),\mathsf{p}(E_2,T),\ldots,\mathsf{p}'(E_m,T))$,
—$\mathsf{p}'(\{E\}_{K_i},T) = \{\Box\}_{K_i}$ if $K_i \notin T$,
—$\mathsf{p}'(\{E\}_{K_i},T) = \{\mathsf{p}'(E,T)\}_{K_i}$ if $K_i \in T$,

where $\Box$ is a special symbol that represents undecryptable material, and

$$\mathsf{pattern}'(E) = \mathsf{p}'(E,\mathsf{recoverable}(E))$$

The equation $\mathsf{p}'(\{E\}_{K_i},T) = \{\Box\}_{K_i}$ (for $K_i \notin T$) reflects the idea that encryption does not reveal anything about the encrypted plaintext. For instance, we have that $\mathsf{pattern}'(\{D_1\}_{K_1}) = \mathsf{pattern}'(\{(\{D_2\}_{K_2},\{D_3\}_{K_3})\}_{K_1})$. An analogue of Theorem 1 holds with this definition. On the other hand, the other theorems of the paper are more problematic in this respect. In particular, an analogue of Theorem 2 does not hold. A stronger security requirement on encryption yields that analogue, and a corresponding strengthening of Theorem 4. That requirement is obtained from the definition of IND-CPA security by removing the restriction that the bit-strings m_0 and m_1 have equal length in queries (m_0,m_1) to the left-right oracle.

Going further, in another formal variation we may replace p and $\mathsf{pattern}$ with the functions p'' and $\mathsf{pattern}''$ defined as follows:

—$\mathsf{p}''(B,T) = B$ for all $B \in \mathsf{BData}$,
—$\mathsf{p}''((E_1,E_2,\ldots,E_m),T) = (\mathsf{p}''(E_1,T),\mathsf{p}(E_2,T),\ldots,\mathsf{p}''(E_m,T))$,
—$\mathsf{p}''(\{E\}_{K_i},T) = \Box$ if $K_i \notin T$,

—$\mathsf{p}''(\{E\}_{K_i}, T) = \{\mathsf{p}''(E, T)\}_{K_i}$ if $K_i \in T$,

thus hiding the occurrence of the key K_i when $\Box$ is produced, and

$$\mathsf{pattern}''(E) = \mathsf{p}''(E, \mathsf{recoverable}(E))$$

The equation $\mathsf{p}''(\{E\}_{K_i}, T) = \Box$ (for $K_i \notin T$) implies that an adversary that does not know the encryption key K_i a priori cannot identify the use of K_i. For instance, we have that $\mathsf{pattern}''((\{D_1\}_{K_1}, \{D_2\}_{K_1}) = \mathsf{pattern}''((\{D_1\}_{K_1}, \{D_2\}_{K_2})$; this equation indicates that an adversary cannot tell whether the two pieces of data D_1 and D_2 are protected by the same key. Such guarantees might matter in settings where the security policy itself is sensitive information. They do not hold with IND-CPA security, so they do not follow from Theorem 4. Again we need a strengthening of IND-CPA security, such as type-0 security [Abadi and Rogaway 2002].

For brevity, we omit precise statements and proofs of the analogues of Theorems 2 and 4 for $\mathsf{pattern}'$ and $\mathsf{pattern}''$.

Attacks with Multiple Queries

Definition 3 is concerned with an adversary that attempts to recover secret information from a single document. In this section, we consider more general settings in which the adversary attempts to extract information from several related documents.

As a motivating scenario, let us consider an electronic journal that offers several types of possible subscriptions. Each type enables access to a selection of journal sections. The access policy can be enforced by means of a protection: articles (and groups of articles) are encrypted and each subscriber receives an appropriate set of decryption keys. Some aspects of this scenario do not seem to be adequately captured by Definition 3. In this scenario, an adversary may have access to several related documents (several issues of the journal), and may influence the contents of one document on the basis of the contents of past ones (for instance, by publishing articles in the journal). User dynamics (new subscriptions and cancelled subscriptions) can cause further complications.

Next we give stronger security definitions, with such scenarios in mind. Then we prove strengthenings of results of Section 4. These strengthenings do not require additional cryptographic hypotheses, and yield security guarantees with respect to more general, more active adversaries.

For our definitions, we introduce an oracle that enables an adversary to request encrypted versions of documents of its choosing. Formally, we let $\mathcal{EKG}(\eta)$ generate vectors of keys and key shares for security parameter η, and we write $[\![E]\!]^{\Pi,\mathcal{SS},\eta}_{\phi,\tau,f_b}$ for the result of the procedure defined in Figure 3 for fixed τ and ϕ; then for expression E, encryption scheme Π, secret sharing scheme $\mathcal{SS}$, and set $S \subseteq \mathsf{Data}$, and for any $(\tau, \phi) \stackrel{R}{\leftarrow} \mathcal{EKG}(\eta)$, we introduce an oracle $O([\![E]\!]^{\Pi,\mathcal{SS}}_{\tau,\phi}, S, \eta, b)$ that expects to receive a pair of valuations (f_0, f_1) that coincide on S and returns a sample from $[\![E]\!]^{\Pi,\mathcal{SS},\eta}_{\phi,\tau,f_b}$. Thus, the oracle O uses a fixed pair (τ, ϕ); our results have analogues for the case in which O generates a fresh pair (τ, ϕ) at each query. The oracle O also uses a fixed expression E; we have yet to study the case in which E may vary under control of the adversary.

Definition 4 is a variant of Definition 3 in which an adversary has access to such an oracle O:

DEFINITION 4. *Let E be an expression, Π an encryption scheme, $\mathcal{SS}$ a secret sharing scheme, and $S \subseteq \mathsf{Data}$. The set S is computationally hidden in E (with Π and $\mathcal{SS}$) against multi-query adversaries if for any probabilistic polynomial-time adversary A the function*

$$\begin{aligned}\mathbf{Adv}^{\text{dist-m}}_{\Pi,\mathcal{SS},E,S}(A,\eta) \;=\; &\Pr\left[(\phi,\tau)\stackrel{R}{\leftarrow}\mathcal{EKG}(\eta) : A^{O([\![E]\!]^{\Pi,\mathcal{SS}}_{\tau,\phi},S,\eta,1)}(\eta)=1\right] - \\ &\Pr\left[(\phi,\tau)\stackrel{R}{\leftarrow}\mathcal{EKG}(\eta) : A^{O([\![E]\!]^{\Pi,\mathcal{SS}}_{\tau,\phi},S,\eta,0)}(\eta)=1\right]\end{aligned}$$

is negligible.

The next theorem is an analogue of Theorem 2. Here, an adversary attempts to distinguish between $[\![E]\!]^{\Pi,\mathcal{SS}}_{\tau,\phi,f}$ and $[\![\mathsf{pattern}(E)]\!]^{\Pi,\mathcal{SS}}_{\tau,\phi,f}$ by adaptively choosing several valuations f, for a randomly chosen pair $(\tau,\phi)\stackrel{R}{\leftarrow}\mathcal{EKG}(\eta)$. The formulation of the theorem relies on an oracle $O(E,\Pi,\mathcal{SS},\tau,\phi,\eta)$ that expects to receive valuations as inputs, and that, on input f, returns a sample from distribution $[\![E]\!]^{\Pi,\mathcal{SS},\eta}_{\tau,\phi,f}$. We call an adversary with access to such an oracle a pattern adversary.

THEOREM 5. *Let E be an acyclic expression. If Π is and* IND-CPA *secure encryption scheme and $\mathcal{SS}$ is a secure secret sharing scheme, then for any probabilistic, polynomial-time pattern adversary A the function*

$$\begin{aligned}\mathbf{Adv}^{\text{dist-pat}}_{\Pi,\mathcal{SS},E}(A,\eta) \;=\; &\Pr\left[(\tau,\phi)\stackrel{R}{\leftarrow}\mathcal{EKG}(\eta) : A^{O(E,\Pi,\mathcal{SS},\tau,\phi,\eta)}(\eta)=1\right] - \\ &\Pr\left[(\tau,\phi)\stackrel{R}{\leftarrow}\mathcal{EKG}(\eta) : A^{O(\mathsf{pattern}(E),\Pi,\mathcal{SS},\tau,\phi,\eta)}(\eta)=1\right]\end{aligned}$$

is negligible.

The proof of this theorem resembles that of Theorem 2, and it is presented in the Appendix.

Using this theorem, we obtain the following result, which relates symbolic secrecy and secrecy with respect to multi-query adversaries.

LEMMA 6. *Let E be an acyclic expression and $T=\{K_1,K_2,\ldots,K_l\}\subseteq\mathsf{keys}(E)$ a set of keys. If Π is an* IND-CPA *encryption scheme and $\mathcal{SS}$ is a secure secret sharing scheme, then $\mathsf{Data}-\mathsf{Acc}_E(T)$ is computationally hidden in $(E,K_1,K_2,\ldots,K_l)$ with Π and $\mathcal{SS}$ against multi-query adversaries.*

In order to prove this theorem, we consider a multi-query adversary A, we let S be the set $\mathsf{Data}-\mathsf{Acc}_E(T)$ and let E' be the expression $(E,K_1,K_2,\ldots,K_l)$, and we aim to show that $\mathbf{Adv}^{\text{dist-m}}_{\Pi,\mathcal{SS},E',S}(A,\eta)$ is negligible. First we construct two pattern adversaries A_0 and A_1. For $b=0,1$, adversary A_b executes A internally, and when A produces a pair of valuations (f_0,f_1) as a query to its oracle, A_b passes f_b to its oracle and forwards the answer to A. The output of A_b is the final output of A. It follows from the construction that for $b=0,1$:

$$\begin{aligned}&\Pr\left[(\tau,\phi)\stackrel{R}{\leftarrow}\mathcal{EKG}(\eta) : A_b^{O(E',\Pi,\mathcal{SS},\tau,\phi,\eta)}(\eta)=1\right] = \\ &\quad\Pr\left[(\tau,\phi)\stackrel{R}{\leftarrow}\mathcal{EKG}(\eta) : A^{O([\![E']\!]^{\Pi,\mathcal{SS}}_{\tau,\phi},S,\eta,b)}(\eta)=1\right]\end{aligned}$$

(Here, the probabilities are over the choice $(\tau,\phi) \stackrel{R}{\leftarrow} \mathcal{EKG}(\eta)$, the coins of adversaries, and those of oracles; for brevity, we omit $(\tau,\phi) \stackrel{R}{\leftarrow} \mathcal{EKG}(\eta)$ from formulas below.) We obtain:

$$
\begin{aligned}
&\mathbf{Adv}^{\text{dist-m}}_{\Pi,\mathcal{SS},E',S}(A,\eta) \\
&= \Pr\left[A^{O(\llbracket E' \rrbracket^{\Pi,\mathcal{SS}}_{\tau,\phi},S,\eta,1)}(\eta)=1\right] - \Pr\left[A^{O(\llbracket E' \rrbracket^{\Pi,\mathcal{SS}}_{\tau,\phi},S,\eta,0)}(\eta)=1\right] \\
&= \Pr\left[A_1^{O(E',\Pi,\mathcal{SS},\tau,\phi,\eta)}(\eta)=1\right] - \Pr\left[A_0^{O(E',\Pi,\mathcal{SS},\tau,\phi,\eta)}(\eta)=1\right] \\
&= \left(\Pr[A_1^{O(E',\Pi,\mathcal{SS},\tau,\phi,\eta)}(\eta)=1] - \Pr\left[A_1^{O(\mathsf{pattern}(E'),\Pi,\mathcal{SS},\tau,\phi,\eta)}(\eta)=1\right]\right) + \\
&\quad \left(\Pr\left[A_1^{O(\mathsf{pattern}(E'),\Pi,\mathcal{SS},\tau,\phi,\eta)}(\eta)=1\right] - \Pr\left[A_0^{O(\mathsf{pattern}(E'),\Pi,\mathcal{SS},\tau,\phi,\eta)}(\eta)=1\right]\right) + \\
&\quad \left(\Pr\left[A_0^{O(\mathsf{pattern}(E'),\Pi,\mathcal{SS},\tau,\phi,\eta)}(\eta)=1\right] - \Pr\left[A_0^{O(E',\Pi,\mathcal{SS},\eta,\tau,\phi,1)}(\eta)=1\right]\right) \\
&= \mathbf{Adv}^{\text{dist-pat}}_{\Pi,\mathcal{SS},E'}(A_1,\eta) + \\
&\quad \left(\Pr\left[A_1^{O(\mathsf{pattern}(E'),\Pi,\mathcal{SS},\tau,\phi,\eta)}(\eta)=1\right] - \Pr\left[A_0^{O(\mathsf{pattern}(E'),\Pi,\mathcal{SS},\tau,\phi,\eta)}(\eta)=1\right]\right) + \\
&\quad -\mathbf{Adv}^{\text{dist-pat}}_{\Pi,\mathcal{SS},E'}(A_0,\eta)
\end{aligned}
$$

By Theorem 5, the first and the last summands are negligible. The middle summand is 0 because A's queries (f_0, f_1) consist of valuations that coincide on the set $\mathsf{Acc}_{E'}(T)$, and (by the definition of $\mathsf{Acc}_{E'}(T)$) these are the only symbols that occur in $\mathsf{pattern}(E')$. We conclude that $\mathbf{Adv}^{\text{dist-m}}_{\Pi,\mathcal{SS},E',S}(A,\eta)$ is negligible.

Finally, Theorem 1 and Lemma 6 immediately yield the following analogue of Theorem 4:

THEOREM 7. *Let P be a normalized protection such that $E(P)$ is an acyclic expression. Let $T = \{K_1, K_2, \ldots, K_l\} \subseteq \mathsf{keys}(E(P))$ be an arbitrary set of keys. If Π is an* IND-CPA *secure encryption scheme and $\mathcal{SS}$ is a secure secret sharing scheme, then $\mathsf{Data} - \mathsf{Acc}_P(T)$ is computationally hidden in $(E(P), K_1, K_2, \ldots, K_l)$ with Π and $\mathcal{SS}$ against multi-query adversaries.*

Attacks with Selective Decryption

We close this section with a brief discussion of security under selective decryption attacks (SD-security for short) [Canetti et al. 1996]. Consider an attacker that receives a set of ciphertexts and then chooses a subset to be decrypted; the choice may depend on the ciphertexts received. The problem addressed by research on SD-security is to show that the remaining, undecrypted plaintexts are still protected.

Generally, formal methods do not give special consideration to selective decryption attacks; basically, selective decryption is not regarded as any more harmful than any other decryption, and it need not endanger undecrypted plaintexts. On the other hand, from a computational perspective, research on selective decommitment [Dwork et al. 2003] (a closely related subject) would suggest that proving the existence of SD-secure encryption schemes would require the development of new cryptographic techniques.

This particular discrepancy between formal and computational views need not be an immediate concern in the present setting, since it need not present the se-

lective decryption problem in full generality. Specifically, in Section 4, the set of encryptions available to an adversary in a single document depends only on the document, and is independent of the security parameter. Under analogous conditions, it is possible to show that standard security implies SD-security for commitment schemes [Dwork et al. 2003], and the proof may well extend to encryption schemes. For more general adversaries (such as multi-query adversaries), other solutions should be sought. One possibility may be to explore non-committing encryption [Canetti et al. 1996], as in the setting of multi-party computation. Unfortunately, at present, non-committing encryption can be quite inefficient.

6. CONCLUSION

The main contribution of this paper is a precise justification of the encryption-based techniques for enforcing access policies for XML documents, as developed by Miklau and Suciu. More specifically, we provide a proof that XML data that is secret according to an abstract, symbolic semantics is indeed secret with respect to a strong, computational notion of security.

In defining the subject of our analysis, we have attempted to be faithful to the work of Miklau and Suciu. In further research, one might like to depart from their framework. In particular, like them, we have focused on protection against off-line attacks. In such attacks, the adversary obtains information about a document only by observing it. However, we have allowed for some interaction with the adversary, enabling the adversary to influence the contents of documents; in further research, it may be interesting to consider more general active attacks. It may also be interesting to consider richer access control policies, including integrity requirements, as well as richer data models with key constraints, functional dependencies, and other refinements. The value of rigorous analysis may be even larger with these enrichments, but our basic approach should remain applicable and helpful in bridging the gap between high-level designs and precise guarantees.

ACKNOWLEDGMENTS

We are grateful to Véronique Cortier, Phokion Kolaitis, Daniele Micciancio, Gerome Miklau, Dan Suciu, and Victor Vianu for helpful discussions on this work and its presentation.

REFERENCES

Abadi, M. and Rogaway, P. 2002. Reconciling two views of cryptography (The computational soundness of formal encryption). *Journal of Cryptology 15,* 2, 103–127.

Adam, N. R. and Worthmann, J. C. 1989. Security-control methods for statistical databases: a comparative study. *ACM Comput. Surv. 21,* 4, 515–556.

Backes, M., Pfitzmann, B., and Waidner, M. 2003. A composable cryptographic library with nested operations. In *10th ACM Conference on Computer and Communications Security*. 220–330. Long version: IACR ePrint Archive, Report 2003/015.

Bellare, M. and Rogaway, P. 2005. Introduction to modern cryptography. Available at: `http://www.cs.ucsd.edu/~mihir/cse207/classnotes.html`.

Bertino, E., Carminati, B., and Ferrari, E. 2002. A temporal key management scheme for secure broadcasting of XML documents. In *8th ACM Conference on Computer and Communications Security*. 31–40.

Bertino, E., Castano, S., and Ferrari, E. 2001. Author-X: A comprehensive system for securing XML documents. *IEEE Internet Computing 5,* 3, 21–31.

Canetti, R., Dwork, C., Goldreich, O., and Naor, M. 1996. Adaptively secure multiparty computation. In *28th ACM Symposium on Theory of Computing*. 639–648.

Castano, S., Fugini, M. G., Martella, G., and Samarati, P. 1995. *Database Security*. Addison-Wesley – ACM Press.

Crampton, J. 2004. Applying hierarchical and role-based access control to XML documents. In *ACM Workshop on Secure Web Services*. 41–50.

Damiani, E., de Capitani di Vimercati, S., Paraboschi, S., and Samarati, P. 2002. A fine-grained access control system for XML documents. *ACM Transactions on Information and System Security 5,* 2, 169–202.

Dolev, D., Dwork, C., and Naor, M. 2000. Non-malleable cryptography. *SIAM Journal of Computing 30,* 2, 391–437.

Dwork, C., Naor, M., Reingold, O., and Stockmeyer, L. J. 2003. Magic functions. *Journal of the ACM 50,* 6, 852–921.

Eastlake, D. and Reagle, J. 2002. XML encryption syntax and processing. `http://www.w3.org/TR/xmlenc-core`.

Gifford, D. K. 1982. Cryptographic sealing for information secrecy and authentication. *Communications of the ACM 25,* 4, 274–286.

Goldwasser, S. and Micali, S. 1984. Probabilistic encryption. *Journal of Computer and System Sciences 28*, 270–299.

Herzog, J. 2004. Computational soundness for standard assumptions of formal cryptography. Ph.D. thesis, Massachusetts Institute of Technology.

Kudo, M. and Hada, S. 2000. XML document security based on provisional authorization. In *7th ACM Conference on Computer and Communications Security*. 87–96.

Laud, P. 2004. Symmetric encryption in automatic analyses for confidentiality against active adversaries. In *2004 IEEE Symposium on Security and Privacy*. 71–85.

Micciancio, D. 2004. Towards computationally sound symbolic security analysis. Talk at DIMACS; slides available at: `http://dimacs.rutgers.edu/Workshops/Protocols/slides/micciancio.pdf`.

Micciancio, D. and Panjwani, S. 2005. Adaptive security of symbolic encryption. In *Theory of Cryptography Conference (TCC 2005)*. Springer-Verlag, 169–187.

Micciancio, D. and Panjwani, S. 2006. Corrupting one vs. corrupting many: The case of broadcast and multicast encryption. In *Automata, Languages and Programming, 33rd International Colloquium, Proceedings, Part II*. Springer-Verlag, 70–82.

Micciancio, D. and Warinschi, B. 2004. Soundness of formal encryption in the presence of active adversaries. In *Theory of Cryptography Conference (TCC 2004)*. Springer-Verlag, 133–151.

Miklau, G. and Suciu, D. 2003. Controlling access to published data using cryptography. In *VLDB 2003: 29th International Conference on Very Large Data Bases*. 898–909.

Shamir, A. 1979. How to share a secret. *Communications of the ACM 22,* 11, 612–613.

Ullman, J. 1983. *Principles of Database Systems*. Computer Science Press, Potomac, MD.

Yang, X. and Li, C. 2004. Secure XML publishing without information leakage in the presence of data inference. In *VLDB 2004: 30th International Conference on Very Large Data Bases*. 96–107.

A. APPENDIX: PROOF OF THEOREM 2

In this appendix we provide a proof of Theorem 2. Some aspects of the proof are by now standard; the reader may wish to consult the proof of the main theorem in the work of Abadi and Rogaway [Abadi and Rogaway 2002]. We start with some notation and useful lemmas.

A permutation $\sigma : \mathsf{Keys} \cup \mathsf{KeyShares} \to \mathsf{Keys} \cup \mathsf{KeyShares}$ is *consistent* if

—$\sigma(K_i) \in \mathsf{Keys}$ for all $K_i \in \mathsf{Keys}$, and
—if $K_i = \sigma(K_j)$, then $K_i^l = \sigma(K_j^l)$ for all $l \in \{1, 2, \ldots, n\}$.

For any consistent permutation σ, we write $E\sigma$ for the expression obtained from E by renaming its keys and key shares according to σ. Consistent renamings do not change the distribution ensembles associated with expressions:

LEMMA 8. *For any expression E, if σ is a consistent permutation, then:*

$$[\![E]\!]_f^{\Pi,\mathcal{SS}} \approx [\![E\sigma]\!]_f^{\Pi,\mathcal{SS}}$$

This lemma holds because the output of the algorithm that associates a distribution with an expression does not depend on the actual atomic symbols used in the expression, but only on their meaning.

For each expression and pattern E, we write $\mathsf{hidden}(E)$ for the set of keys that are not recoverable from E:

$$\mathsf{hidden}(E) = \mathsf{keys}(E) - \mathsf{recoverable}(E)$$

The following lemma states that the keys in any acyclic expression E can be reordered so that the first l keys are the keys hidden in E and the remaining keys are the keys that can be recovered from E. Moreover, it is possible to reorder the keys so that, in the resulting expression, for any two hidden keys K_i and K_j, if K_i encrypts K_j then $i < j$.

LEMMA 9. *If E is an acyclic expression, $m = |\mathsf{keys}(E)|$, and $l = |\mathsf{hidden}(E)|$, then there exists a consistent permutation σ such that:*

—$\mathsf{keys}(E\sigma) = \{K_1, K_2, \ldots, K_l, K_{l+1}, \ldots, K_m\}$,
—$\mathsf{hidden}(E\sigma) = \{K_1, K_2, \ldots, K_l\}$,
—*for any $1 \le i, j \le l$, if K_i encrypts K_j in $E\sigma$ then $i < j$.*

By Lemmas 8 and 9, it is sufficient to prove the theorem under the following additional assumptions:

—$\mathsf{keys}(E) = \{K_1, K_2, \ldots, K_l, K_{l+1}, \ldots, K_m\}$,
—$\mathsf{hidden}(E) = \{K_1, K_2, \ldots, K_l\}$,
—for any $1 \le i, j \le l$, if K_i encrypts K_j then $i < j$.

Proof of the Theorem

We prove the theorem by a hybrid argument: given an expression E as above, we exhibit a series of distribution ensembles $\mathcal{D}^0, \mathcal{D}^1, \ldots, \mathcal{D}^l$, such that:

(1) $\mathcal{D}^0 = [\![E]\!]_f^{\Pi,\mathcal{SS}}$,
(2) $\mathcal{D}^l = [\![\mathsf{pattern}(E)]\!]_f^{\Pi,\mathcal{SS}}$, and
(3) $\mathcal{D}^{i-1} \approx \mathcal{D}^i$, for all $1 \le i \le l$.

Since l is a fixed constant (independent of the security parameter), the conclusion of the theorem immediately follows.

Consider the sequence of patterns E_0, E_1, …, E_l inductively defined by:

—$E_0 = E$,

—E_i is obtained by replacing each subexpression of E_{i-1} of the form $\{E'\}_{K_i}$ with $\{\mathsf{struct}(E')\}_{K_i}$.

For each $0 \leq i \leq l$, we let $\mathcal{D}^i$ be the distribution ensemble associated with pattern E_i and prove that the resulting sequence of distribution ensembles satisfies conditions 1–3. It is immediate that $\mathcal{D}^0 = [\![E]\!]_f^{\Pi,\mathcal{SS}}$. Also, since pattern E_l is obtained by replacing all subexpressions of E of the form $\{E'\}_{K_i}$ for some $K_i \in \mathsf{hidden}(E)$ with $\{\mathsf{struct}(E')\}_{K_i}$, we have that $E_l = \mathsf{pattern}(E)$ and, therefore, that $\mathcal{D}^l = [\![\mathsf{pattern}(E)]\!]_f^{\Pi,\mathcal{SS}}$. It only remains to be shown that, for each $1 \leq i \leq l$, the distribution ensembles $\mathcal{D}^{i-1}$ and $\mathcal{D}^i$ are indistinguishable (condition 3). For each $0 \leq i \leq l-1$, we introduce two intermediate distribution ensembles $\mathcal{D}^{i,0}$ and $\mathcal{D}^{i,1}$ and we prove that:

$$\mathcal{D}^i \approx \mathcal{D}^{i,0} \approx \mathcal{D}^{i,1} \approx \mathcal{D}^{i+1}$$

It follows that $\mathcal{D}^i \approx \mathcal{D}^{i+1}$ for all $0 \leq i \leq l-1$, and we can conclude that

$$[\![E]\!]_f^{\Pi,\mathcal{SS}} = \mathcal{D}^0 \approx \mathcal{D}^l = [\![\mathsf{pattern}(E)]\!]_f^{\Pi,\mathcal{SS}}$$

Consider the patterns $E_{i,0}$ and $E_{i,1}$ defined as follows:

—$E_{i,0}$ is obtained from E_i by replacing each occurrence of a key share symbol K_{i+1}^j with a fresh key share symbol K^j, different from those in $\mathsf{KeyShares} \cup \{K_0^j \mid j \in 1..n\}$.

—$E_{i,1}$ is obtained from $E_{i,0}$ by replacing each occurrence of a subexpression of the form $\{E'\}_{K_{i+1}}$ with the corresponding expression $\{\mathsf{struct}(E')\}_{K_{i+1}}$.

It follows from the description above that E_{i+1} can be obtained from $E_{i,1}$ by replacing each occurrence of a key share K^j with K_{i+1}^j.

Next we associate distributions (and therefore distribution ensembles) with the patterns $E_{i,0}$ and $E_{i,1}$ via a slight modification of the algorithm of Section 4. Specifically, we introduce computational interpretations for the newly introduced symbols $K^1, K^2, \ldots, K^n$: we add one new position to the array τ and one extra row to the matrix ϕ and set:

$$\tau[m+1] \stackrel{R}{\leftarrow} \mathcal{K}(\eta)$$
$$\phi[m+1] \stackrel{R}{\leftarrow} \mathcal{S}(\tau[m+1], \eta)$$

We use the bit-strings contained in $\phi[m+1]$ as interpretations of the symbols K^1, K^2, …, K^n, so we add to the algorithm in Figure 3 the line:

$$[\![K^j]\!]_f^{\Pi,\mathcal{SS},\eta} = \phi[m+1][j]$$

For each $0 \leq i \leq l-1$, we let $\mathcal{D}^{i,0}$ and $\mathcal{D}^{i,1}$ be the distribution ensembles associated with $E_{i,0}$ and $E_{i,1}$.

The remainder of the proof consists of three steps. They respective establish that $\mathcal{D}^i \approx \mathcal{D}^{i,0}$, that $\mathcal{D}^{i,0} \approx \mathcal{D}^{i,1}$, and that $\mathcal{D}^{i,1} \approx \mathcal{D}^{i+1}$.

Step 1 ($\mathcal{D}^i \approx \mathcal{D}^{i,0}$). The proof is by reduction. Given an index $0 \leq h \leq l-1$ and an algorithm A such that $\mathbf{Adv}^{\mathrm{dist}}_{\mathcal{D}^h,\mathcal{D}^{h,0}}(A,\eta)$ is non-negligible, we construct an adversary B against $\mathcal{SS}$ such that $\mathbf{Adv}^{\mathrm{ss}}_{\mathcal{SS}}(B,\eta)$ is also non-negligible. Therefore, the scheme $\mathcal{SS}$ is not a secure secret sharing scheme.

In the construction of B we use the set

$$S_h = \{j \mid K_{h+1}^j \text{ occurs in } E_h\}$$

of indexes j for which the key share K_{h+1}^j occurs in E_h. The occurrences of K_{h+1}^j in question cannot be under hidden keys, because of the acyclicity hypothesis. Therefore, crucially, S_h is a proper subset of $\{1, 2, \ldots, n\}$, for otherwise the key K_{h+1} would not be in $\mathsf{hidden}(E)$.

The adversary B that we construct receives as input two keys $k_0, k_1 \stackrel{R}{\leftarrow} \mathcal{K}(\eta)$ and a set of shares $(s_1, s_2, \ldots, s_t)$ sampled according to one of the two distributions $\mathcal{S}(k_0, \eta)|_{S_h}$ or $\mathcal{S}(k_1, \eta)|_{S_h}$. It constructs a string s so that if $(s_1, s_2, \ldots, s_t)$ is sampled according to the distribution $\mathcal{S}(k_0, \eta)|_{S_h}$, then s is sampled according to $\mathcal{D}^h$, and if $(s_1, s_2, \ldots, s_t)$ is sampled according to the distribution $\mathcal{S}(k_1, \eta)|_{S_h}$, then s is sampled according to $\mathcal{D}^{h,0}$. Adversary B then invokes the algorithm A on input s and outputs whatever A outputs. We therefore obtain that:

$$\begin{aligned}
&\mathbf{Adv}^{\mathrm{ss}}_{\mathcal{SS}}(B, \eta) \\
&= \Pr\left[k_0, k_1 \stackrel{R}{\leftarrow} \mathcal{K}(\eta), sh(k_0) \stackrel{R}{\leftarrow} \mathcal{S}(k_0, \eta) : B(k_0, k_1, sh(k_0)|_{S_h}) = 1)\right] - \\
&\quad \Pr\left[k_0, k_1 \stackrel{R}{\leftarrow} \mathcal{K}(\eta), sh(k_1) \stackrel{R}{\leftarrow} \mathcal{S}(k_1, \eta) : B(k_0, k_1, sh(k_1)|_{S_h}) = 1)\right] \\
&= \Pr\left[s \stackrel{R}{\leftarrow} [\![E_h]\!]_f^{\Pi, \mathcal{SS}, \eta} : A(s, \eta) = 1\right] - \Pr\left[s \stackrel{R}{\leftarrow} [\![E_{h,0}]\!]_f^{\Pi, \mathcal{SS}, \eta} : A(s, \eta) = 1\right] \\
&= \mathbf{Adv}^{\mathrm{dist}}_{\mathcal{D}^h, \mathcal{D}^{h,0}}(A, \eta)
\end{aligned}$$

It follows that if the advantage of A is non-negligible then so is that of B.

Adversary B computes s by applying to E_h a variant of the algorithm given in Section 4 for mapping expressions to bit-strings. Specifically, B starts by generating the vector τ of keys in which the entries $\tau[h+1]$ and $\tau[m+1]$ are set to k_0 and k_1. Next, B computes the entries in the matrix ϕ (used for defining the semantics of the key shares that occur in E_h). Both τ and ϕ have the appropriate distributions since k_0 and k_1 are randomly generated keys.

Then B computes the string s by recursively associating bit-strings with the subexpressions of E_h. The only departure from the algorithm of Section 4 is as follows. The bit-strings associated with the shares of K_{h+1} are interpreted using the input to B. (In the original algorithm, the bit-string interpretation of these shares are the entries in $\phi[h+1]$, that is, shares of $\tau[h+1]$.) So, if $S_h = \{j_1, j_2, \ldots, j_t\}$, then the modified algorithm maps K_{h+1}^p to s_p, for each $1 \leq p \leq t$. Crucially, if $(s_1, s_2, \ldots, s_t)$ is sampled according to $sh(k_0)|_{S_h}$, then the key K_{h+1} is mapped to k_0 and all shares $K_{h+1}^{j_p}$ are mapped to appropriate shares of k_0. Therefore, the string s is selected according to the distribution $[\![E_h]\!]_f^{\Pi, \mathcal{SS}, \eta}$. On the other hand, if $(s_1, s_2, \ldots, s_t)$ is sampled according to $sh(k_1)|_{S_h}$, then although K_{h+1} is mapped to k_0, all its shares are mapped to shares of k_1—that is, to shares of a different key. Therefore, in this case, s is selected according to the distribution $[\![E_{h,0}]\!]_f^{\Pi, \mathcal{SS}, \eta}$, as desired.

Step 2 ($\mathcal{D}^{i,0} \approx \mathcal{D}^{i,1}$). The proof is again by reduction. Given an index $0 \leq h \leq l-1$ and an algorithm A that distinguishes between $\mathcal{D}^{h,0}$ and $\mathcal{D}^{h,1}$ with non-negligible probability, we construct an adversary B against Π such that $\mathbf{Adv}^{\text{ind-cpa}}_{\Pi}(B, \eta)$ is

also non-negligible. The adversary B has access to the oracle $LR_{\Pi,b}$ and constructs a bit-string s that is sampled according to the distribution $\mathcal{D}^{h,b}$. Then B invokes algorithm A on input s and outputs whatever A outputs. We thus obtain:

$$\begin{aligned}
&\mathbf{Adv}^{\text{ind-cpa}}_{\Pi}(B,\eta)\\
&= \Pr[B^{LR_{\Pi,0}(\eta)}(\eta)=1] - \Pr[B^{LR_{\Pi,1}(\eta)}(\eta)=1]\\
&= \Pr[s \stackrel{R}{\leftarrow} \mathcal{D}^{h,0} : A(s)=1] - \Pr[s \stackrel{R}{\leftarrow} \mathcal{D}^{h,1} : A(s)=1]\\
&= \mathbf{Adv}^{\text{dist}}_{\mathcal{D}^{h,0},\mathcal{D}^{h,1}}(A,\eta)
\end{aligned}$$

It follows that if the advantage of A is non-negligible then so is that of B. Therefore, the scheme Π is not an IND-CPA secure encryption scheme.

Adversary B computes s by applying to $E_{h,0}$ a variant of the algorithm given in Section 4. Specifically, B generates the vector of keys τ and the matrix of key shares ϕ and then recursively maps each subexpression F of $E_{h,0}$ to a bit-string. The interpretations of all basic symbols with the exception of the key K_{h+1} are as in Section 4. The interpretation of K_{h+1} is set to k, the key of the oracle. Since B does not actually have k, it is crucial that no share of K_{h+1} occurs in $E_{h,0}$. By acyclicity, there are no plain occurrences of K_{h+1} in $E_{h,0}$ either. There may however be uses of K_{h+1} as an encryption key. In order to deal with those occurrences, B makes uses of the oracle for producing encryptions under k, as follows.

Suppose that $E_{h,0}$ has a subexpression F of the form $\{E'\}_{K_{h+1}}$. First, B samples strings m_0 and m_1 from the distributions associated with E' and $\mathsf{struct}(E')$, respectively. This task is mostly straightforward since B knows the interpretations of all symbols that occur in E' and $\mathsf{struct}(E')$ with the exception of the interpretation of K_{h+1}. Whenever B needs to compute an encryption of a plaintext m under the key k (which is the interpretation of K_{h+1}), B submits to the oracle the pair (m,m) and obtains in return one such encryption. After producing m_0 and m_1 as described above, B submits to the oracle the pair (m_0,m_1) and sets c to be the answer returned by the oracle. Therefore, if the selection bit of the oracle is 0, then c is an encryption of m_0 under k, so c is distributed according to $[\![\{E'\}_{K_{h+1}}]\!]_f^{\Pi,\mathcal{SS},\eta}$. If the selection bit of the oracle is 1, then c is an encryption of m_1 under k, so c is distributed according to $[\![\{\mathsf{struct}(E')\}_{K_{h+1}}]\!]_f^{\Pi,\mathcal{SS},\eta}$. Since $E_{i,1}$ is obtained by replacing in $E_{i,0}$ each subexpression F of the form $\{E'\}_{K_{i+1}}$ with $\{\mathsf{struct}(E')\}_{K_{i+1}}$, the overall result of B's computation is distributed like the interpretation of $E_{i,b}$.

Importantly, B is a valid IND-CPA adversary: each query (m_0,m_1) that B makes to its right-left oracle is valid since strings m_0 and m_1 are sampled according to distributions $[\![E']\!]^{\Pi,\mathcal{SS},\eta}$ and $[\![\mathsf{struct}(E')]\!]^{\Pi,\mathcal{SS},\eta}$, respectively, and therefore have equal lengths (by the assumption on the implementation discussed in Section 4).

Step 3 ($\mathcal{D}^{i,1} \approx \mathcal{D}^{i+1}$). This step is similar to the proof that $\mathcal{D}^{i} \approx \mathcal{D}^{i,0}$: one can think of $E_{i,1}$ as obtained from E_{i+1} by replacing the key share symbols K^j_{i+1} with fresh key share symbols K^j, that is, in precisely the same manner in which $E_{i,0}$ is obtained from E_i. The same proof method applies.

B. APPENDIX: PROOF OF THEOREM 5

Since Theorem 5 is an extension of Theorem 2, the two proofs have the same structure, and they share many ideas. While Theorem 2 could be obtained as a

corollary of Theorem 5, we prefer to present its proof separately, for clarity, and then to omit a few common details in the following proof of Theorem 5.

Proof of the Theorem

By Lemmas 8 and 9 it is sufficient to prove the theorem under the additional assumptions that:

—$\mathsf{keys}(E) = \{K_1, K_2, \ldots, K_l, K_{l+1}, \ldots, K_m\}$,
—$\mathsf{hidden}(E) = \{K_1, K_2, \ldots, K_l\}$,
—for any $1 \leq i, j \leq l$, if K_i encrypts K_j then $i < j$.

We define a series of oracles, $O_0(\tau, \phi, \eta), O_2(\tau, \phi, \eta), \ldots, O_l(\tau, \phi, \eta)$ such that the following conditions hold for any $(\tau, \phi) \stackrel{R}{\leftarrow} \mathcal{EKG}(\eta)$:

(1) the behaviors of oracles $O_0(\tau, \phi, \eta)$ and $O(E, \Pi, \mathcal{SS}, \tau, \phi, \eta)$ are identical,
(2) the behavior of oracles $O_l(\tau, \phi, \eta)$ and $O(\mathsf{pattern}(E), \Pi, \mathcal{SS}, \tau, \phi, \eta)$ are identical,
(3) for all $i \in \{0, 1, \ldots, l-1\}$, no probabilistic polynomial-time pattern adversary A can distinguish whether it interacts with oracle O_i or with oracle O_{i+1}. Formally, for all $0 \leq i \leq l-1$ the function

$$\begin{aligned}\mathbf{Adv}^{\mathrm{dist}}_{O_i, O_{i+1}}(A, \eta) \;=\; & \Pr\left[(\tau, \phi) \stackrel{R}{\leftarrow} \mathcal{EKG}(\eta) : A^{O_i(\tau,\phi,\eta)}(\eta) = 1\right] - \\ & \Pr\left[(\tau, \phi) \stackrel{R}{\leftarrow} \mathcal{EKG}(\eta) : A^{O_{i+1}(\tau,\phi,\eta)}(\eta) = 1\right]\end{aligned}$$

is negligible.

The conclusion of the theorem follows since:

$$\begin{aligned}\mathbf{Adv}^{\text{dist-pat}}_{\Pi, \mathcal{SS}, E}(A, \eta) \;=\; & \Pr\left[(\tau, \phi) \stackrel{R}{\leftarrow} \mathcal{EKG}(\eta) : A^{O(E,\Pi,\mathcal{SS},\tau,\phi,\eta)}(\eta) = 1\right] - \\ & \Pr\left[(\tau, \phi) \stackrel{R}{\leftarrow} \mathcal{EKG}(\eta) : A^{O(\mathsf{pattern}(E),\Pi,\mathcal{SS},\tau,\phi,\eta)}(\eta) = 1\right] \\ =\; & \sum_{i=0}^{l-1} \left(\begin{array}{l} \Pr\left[(\tau, \phi) \stackrel{R}{\leftarrow} \mathcal{EKG}(\eta) : A^{O_i(\tau,\phi,\eta)}(\eta) = 1\right] - \\ \\ \Pr\left[(\tau, \phi) \stackrel{R}{\leftarrow} \mathcal{EKG}(\eta) : A^{O_{i+1}(\tau,\phi,\eta)}(\eta) = 1\right] \end{array} \right) \\ =\; & \sum_{i=0}^{l-1} \mathbf{Adv}^{\mathrm{dist}}_{O_i, O_{i+1}}(A, \eta)\end{aligned}$$

Since the sum of a constant number of negligible functions is negligible, we conclude that $\mathbf{Adv}^{\text{dist-pat}}_{\Pi, \mathcal{SS}, E}(A, \eta)$ is negligible, as desired.

Consider the sequence of patterns E_0, E_1, …, E_l inductively defined as in the proof of Theorem 2. For $0 \leq i \leq l$, we let O_i be $O(E_i, \Pi, \mathcal{SS}, \tau, \phi, \eta)$. By construction, $E_0 = E$ and $E_l = \mathsf{pattern}(E)$, so conditions 1 and 2 above are clearly satisfied.

The next step is to show that, for any probabilistic-polynomial time adversary A, the function $\mathbf{Adv}^{\mathrm{dist}}_{O_i, O_{i+1}}(A, \eta)$ is negligible (condition 3). For each $i \in \{0, 1, \ldots, l-1\}$, we introduce two intermediate oracles $O_{i,0}(\tau, \phi, \eta)$ and $O_{i,1}(\tau, \phi, \eta)$. We let $O_{i,0}(\tau, \phi, \eta)$ be $O(E_{i,0}, \Pi, \mathcal{SS}, \tau, \phi, \eta)$ and let $O_{i,1}(\tau, \phi, \eta)$ be $O(E_{i,1}, \Pi, \mathcal{SS}, \tau, \phi, \eta)$,

where $E_{i,0}$ and $E_{i,1}$ are as in the proof of Theorem 2. We prove that, for $0 \leq i \leq l-1$ and any probabilistic polynomial-time adversary A, the functions:

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{dist}}_{O_i,O_{i,0}}(A,\eta) &= \Pr\left[(\tau,\phi) \stackrel{R}{\leftarrow} \mathcal{EKG}(\eta) : A^{O_i(\tau,\phi,\eta)}(\eta) = 1\right] - \\
&\quad \Pr\left[(\tau,\phi) \stackrel{R}{\leftarrow} \mathcal{EKG}(\eta) : A^{O_{i,0}(\tau,\phi,\eta)}(\eta) = 1\right] \\
\mathbf{Adv}^{\mathrm{dist}}_{O_{i,0},O_{i,1}}(A,\eta) &= \Pr\left[(\tau,\phi) \stackrel{R}{\leftarrow} \mathcal{EKG}(\eta) : A^{O_{i,0}(\tau,\phi,\eta)}(\eta) = 1\right] - \\
&\quad \Pr\left[(\tau,\phi) \stackrel{R}{\leftarrow} \mathcal{EKG}(\eta) : A^{O_{i,1}(\tau,\phi,\eta)}(\eta) = 1\right] \\
\mathbf{Adv}^{\mathrm{dist}}_{O_{i,1},O_{i+1}}(A,\eta) &= \Pr\left[(\tau,\phi) \stackrel{R}{\leftarrow} \mathcal{EKG}(\eta) : A^{O_{i,1}(\tau,\phi,\eta)}(\eta) = 1\right] - \\
&\quad \Pr\left[(\tau,\phi) \stackrel{R}{\leftarrow} \mathcal{EKG}(\eta) : A^{O_{i+1}(\tau,\phi,\eta)}(\eta) = 1\right]
\end{aligned}$$

are all negligible, so

$$\mathbf{Adv}^{\mathrm{dist}}_{O_i,O_{i+1}}(\eta) = \mathbf{Adv}^{\mathrm{dist}}_{O_i,O_{i,0}}(\eta) + \mathbf{Adv}^{\mathrm{dist}}_{O_{i,0},O_{i,1}}(\eta) + \mathbf{Adv}^{\mathrm{dist}}_{O_{i,1},O_{i+1}}(\eta)$$

is also negligible, as desired. Each of these functions is treated in an argument analogous to one of Steps 1, 2, and 3 in the proof of Theorem 2.

Step 1 ($\mathbf{Adv}^{\mathrm{dist}}_{O_i,O_{i,0}}(\eta)$ is negligible). The proof is by reduction. Given an index $0 \leq h \leq l-1$ and an algorithm A such that $\mathbf{Adv}^{\mathrm{dist}}_{O_h,O_{h,0}}(A,\eta)$ is non-negligible, we construct an adversary B against $\mathcal{SS}$ so that $\mathbf{Adv}^{\mathrm{ss}}_{\mathcal{SS}}(B,\eta)$ is also non-negligible. We let S_h be the set of indexes for the shares of K_{h+1} that occur in E_h, as in the proof of Theorem 2. Since adversary B is against the secret sharing scheme, it receives as input two keys $k_0, k_1 \stackrel{R}{\leftarrow} \mathcal{K}(\eta)$ and a set of shares $(s_1, s_2, \ldots, s_t)$ sampled according to one of the two distributions $\mathcal{S}(k_0,\eta)|_{S_h}$ or $\mathcal{S}(k_1,\eta)|_{S_h}$.

The idea behind the construction of B is to execute A as a subroutine in such a way that, if $(s_1, s_2, \ldots, s_t)$ is sampled according to $\mathcal{S}(k_0,\eta)|_{S_h}$, then A's view is as in its interaction with $O_h(\tau,\phi,\eta)$, while if $(s_1, s_2, \ldots, s_t)$ is sampled according to $\mathcal{S}(k_1,\eta)|_{S_h}$, then A's view is as in its interaction with $O_{h,0}(\tau,\phi,\eta)$, for some $(\tau,\phi) \stackrel{R}{\leftarrow} \mathcal{EKG}(\eta)$ which is kept fixed throughout A's execution in both cases. To every query f from A, B returns a string s sampled according to the distribution $[\![E_h]\!]^{\Pi,\mathcal{SS},\eta}_{\tau,\phi,f}$ (in the former case) or $[\![E_{h,0}]\!]^{\Pi,\mathcal{SS},\eta}_{\tau,\phi,f}$ (in the latter case). Furthermore, whenever A outputs 1, B outputs 1. We therefore obtain that:

$$\begin{aligned}
&\mathbf{Adv}^{\mathrm{ss}}_{\mathcal{SS}}(B,\eta) \\
&= \Pr\left[k_0, k_1 \stackrel{R}{\leftarrow} \mathcal{K}(\eta), sh(k_0) \stackrel{R}{\leftarrow} \mathcal{S}(k_0,\eta) : B(k_0,k_1,sh(k_0)|_{S_h}) = 1)\right] - \\
&\quad \Pr\left[k_0, k_1 \stackrel{R}{\leftarrow} \mathcal{K}(\eta), sh(k_1) \stackrel{R}{\leftarrow} \mathcal{S}(k_1,\eta) : B(k_0,k_1,sh(k_1)|_{S_h}) = 1)\right] \\
&= \Pr\left[(\tau,\phi) \stackrel{R}{\leftarrow} \mathcal{EKG}(\eta) : A^{O_i(\phi,\tau,\eta)}(\eta) = 1\right] - \\
&\quad \Pr\left[(\tau,\phi) \stackrel{R}{\leftarrow} \mathcal{EKG}(\eta) : A^{O_{i,0}(\phi,\tau,\eta)}(\eta) = 1\right] \\
&= \mathbf{Adv}^{\mathrm{dist}}_{O_h,O_{h,0}}(A,\eta)
\end{aligned}$$

First, B generates a vector of keys τ to be used as interpretation for the key symbols that occur in E. The entries $\tau[h+1]$ and $\tau[m+1]$ are set to k_0 and k_1

respectively. Then B computes the entries in matrix ϕ via the sharing algorithm of $\mathcal{SS}$. The resulting (ϕ, τ) is distributed according to $\mathcal{EKG}(\eta)$.

In order to answer a query f from A, B computes a string s by recursively associating a bit-string with the subexpressions of E_h. The difference from the algorithm of Section 4 is that the shares of K_{h+1} are interpreted using the input to B. These shares are shares of either $\tau[h+1]$ or $\tau[m+1]$.

Next, we reason as in Step 1 of the proof of Theorem 2. If $(s_1, s_2, \ldots, s_t)$ is sampled according to $\mathcal{S}(k_0, \eta)|_{S_h}$, then key K_{h+1} is mapped to k_0 and its shares to appropriate shares of k_0, so the string s is sampled according to $[\![E_h]\!]^{\Pi,\mathcal{SS},\eta}_{\tau,\phi,f}$. The rest of A's queries are answered using the values (τ, ϕ) generated at the beginning of B's execution. We conclude that, in this case, A's view is as in its interaction with oracle $O(E_h, \Pi, \mathcal{SS}, \tau, \phi, \eta)$. Similarly, if $(s_1, s_2, \ldots, s_t)$ is sampled according to $\mathcal{S}(k_1, \eta)|_{S_h}$, then K_{h+1} is mapped to k_0 and its shares are mapped to corresponding shares of k_1 (that is, shares of $\tau[m+1]$). Therefore, s is sampled according to $[\![E_{h,0}]\!]^{\Pi,\mathcal{SS},\eta}_{\tau,\phi,f}$, and thus A's view is as in its interaction with oracle $O(E_{h,0}, \Pi, \mathcal{SS}, \tau, \phi, \eta)$.

Step 2 ($\mathbf{Adv}^{\text{dist}}_{O_{h,0},O_{h,1}}(\eta)$ is negligible). The proof is again by reduction. Given an index $0 \leq h \leq l-1$ and adversary A such that

$$\begin{aligned}\mathbf{Adv}^{\text{dist}}_{O_{h,0},O_{h,1}} \;=\; &\Pr\left[(\tau,\phi) \xleftarrow{R} \mathcal{EKG}(\eta) : A^{O_{h,0}(\tau,\phi,\eta)}(\eta) = 1\right] - \\ &\Pr\left[(\tau,\phi) \xleftarrow{R} \mathcal{EKG}(\eta) : A^{O_{h,1}(\tau,\phi,\eta)}(\eta) = 1\right]\end{aligned}$$

is non-negligible we construct an adversary B so that $\mathbf{Adv}^{\text{ind-cpa}}_{\Pi}(B, \eta)$ is also non-negligible. Since adversary B is against encryption scheme Π, it has access to a left-right oracle $LR_{\Pi,b}(\eta)$.

The idea behind the construction of B is to execute A as a subroutine in such a way that A's view is as in interaction with oracle $O(E_{h,b}, \Pi, \mathcal{SS}, \tau, \phi, \eta)$, for some $(\tau, \phi) \xleftarrow{R} \mathcal{EKG}(\eta)$ which is kept fixed throughout A's execution. To every query f from A, B returns a string s sampled according to the distribution $[\![E_{h,b}]\!]^{\Pi,\mathcal{SS},\eta}_{\tau,\phi,f}$. Furthermore, whenever A outputs 1, B outputs 1. Therefore, we obtain:

$$\begin{aligned}&\mathbf{Adv}^{\text{ind-cpa}}_{\Pi}(B, \eta) \\ &= \Pr\left[B^{LR_{\Pi,0}(\eta)}(\eta) = 1\right] - \Pr\left[B^{LR_{\Pi,1}(\eta)}(\eta) = 1\right] \\ &= \Pr\left[(\tau,\phi) \xleftarrow{R} \mathcal{EKG}(\eta) : A^{O(E_{h,0},\Pi,\mathcal{SS},\tau,\phi,\eta)}(\eta) = 1\right] - \\ &\quad \Pr\left[(\tau,\phi) \xleftarrow{R} \mathcal{EKG}(\eta) : A^{O(E_{h,1},\Pi,\mathcal{SS},\tau,\phi,\eta)}(\eta) = 1\right] \\ &= \mathbf{Adv}^{\text{dist}}_{O_{h,0},O_{h,1}}(A, \eta)\end{aligned}$$

First, B generates a vector τ of keys for encryption and the matrix ϕ of corresponding key shares. The key $\tau[h+1]$ is set to be the key k of B's left-right oracle. Crucially, B does not actually need to know k: the shares of K_{h+1} do not occur in $E_{h,0}$ (by construction of $E_{h,0}$), and K_{h+1} itself occurs only as an encryption key. The corresponding encryptions are computed by B's oracle.

In order to answer a query f from A, B recursively maps each subexpression F of $E_{h,0}$ to a bit-string. As in the algorithm described in Section 4, basic symbols are

mapped to bit-strings. The interesting case is that of subexpressions where K_{h+1} occurs. As explained above, these expressions contain K_{h+1} only as encryption key and B deals with this case by using its left-right oracle.

Specifically, the string associated to subexpression $\{E'\}_{K_{h+1}}$ is computed as follows. First, B samples two strings m_0 and m_1 from the distributions associated to E' and $\mathsf{struct}(E')$. As explained in the proof of Theorem 2, this process is rather straightforward. Then B submits to its oracle the pair (m_0, m_1) and obtains in return some bit-string c. The string associated with $\{E'\}_{K_{h+1}}$ is set to c. The distribution of c depends on the bit b of the left-right oracle. When $b = 0$, string c is an encryption of m_0 under k, and is therefore sampled according to distribution $[\![\{E'\}_{K_{h+1}}]\!]^{\Pi,\mathcal{SS},\eta}_{\tau,\phi,f}$. Otherwise (that is, when $b = 1$), string c is an encryption of m_1 under k, and is therefore sampled according to distribution $[\![\{\mathsf{struct}(E')\}_{K_{h+1}}]\!]^{\Pi,\mathcal{SS},\eta}_{\tau,\phi,f}$. Since $E_{h,1}$ is obtained from $E_{h,0}$ by replacing all subterms of the form $\{E'\}_{K_{h+1}}$ with $\{\mathsf{struct}(E')\}_{K_{h+1}}$, it follows that the string s obtained at the end of the procedure satisfies the desired condition, that is, it is sampled according to $[\![E_{h,b}]\!]^{\Pi,\mathcal{SS},\eta}_{\tau,\phi,f}$. In the rest of the execution, B answers A's queries following the procedure described above, and using (τ, ϕ) generated at the beginning of its execution. We conclude that A's view is as in its interaction with oracle $O(E_{h,b}, \Pi, \mathcal{SS}, \tau, \phi, \eta)$, as desired.

Step 3 ($\mathbf{Adv}^{\mathrm{dist}}_{O_{i,1},O_{i+1}}(\eta)$ is negligible). This step is similar to Step 1: expression $E_{i,1}$ may be obtained from expression E_{i+1} by replacing each share K^i_{h+1} of K_{h+1} with share K^i of a fresh key K, that is, in precisely the same manner in which $E_{i,0}$ is obtained from E_i. The same proof method applies.