

Privacy-Preserving Inconsistency Measurement

Carl Corea¹, Timotheus Kampik², and Nico Potyka³

¹ University of Koblenz, Germany

² Umeå University, Sweden; SAP, Germany

³ Cardiff University, UK

ccorea@uni-koblenz.de, tkampik@cs.umu.se, PotykaN@cardiff.ac.uk

Abstract. We investigate a new form of (privacy-preserving) inconsistency measurement for multi-party communication. Intuitively, for two knowledge bases K_A, K_B (of two agents A, B), our results allow to quantitatively assess the degree of inconsistency for $K_A \cup K_B$ without having to reveal the actual contents of the knowledge bases. Using secure multi-party computation (SMPC) and cryptographic protocols, we develop two concrete methods for this use-case and show that they satisfy important properties of SMPC protocols – notably, *input privacy*, i.e., jointly computing the inconsistency degree without revealing the inputs.

1 Introduction

In multi-agent systems, agents may have to cooperate without being allowed to share their internal knowledge with each other. For example, revealing internal knowledge or beliefs of an agent may violate (external) privacy requirements, or agents are both cooperating and competing and do not want to reveal knowledge that may give others a competitive advantage.

In our work, we consider multi-agent systems where the agents carry internal knowledge or beliefs in form of propositional logic knowledge bases (KBs), and assume agents may not be allowed to reveal the contents (i.e., formulas) of their KBs to each other. Still, in order to assess the ability to cooperate, it may be necessary for the agents to verify whether, or to what extent, the knowledge bases are *consistent* with each other. Consider the following simplified example from the financial domain:

Example 1. Consider two agents A and B , with (own) propositional logic KBs K_A and K_B (with agents having knowledge on credit applications, and customers can have different statuses, creditworthiness, or be on a ban list), with:

$$K_A = \{\neg(\text{banList} \wedge \text{creditWorthy})\}$$

$$K_B = \{\text{platinumStatus}; \text{platinumStatus} \rightarrow \text{creditWorthy}; \text{banList}\}$$

Clearly, $K_A \cup K_B$ is inconsistent (we will define inconsistency later), and knowing this may be crucial for the agents. But as stated, we assume A and B do not want to reveal the formulas in their KBs. To solve this issue, we present a novel approach for, what we call, *privacy-preserving inconsistency measurement*. The core idea is that we build on cryptographic protocols from the field of

secure multi-party computation, which allow multiple agents to jointly compute a function $f(K_A, K_B)$ without revealing K_A and K_B . Here, we propose algorithms to compute (as f) different *inconsistency measures* [12] for $K_A \cup K_B$.

Our results allow the agents—in a privacy-preserving way—to know i) whether their knowledge is *consistent*, and ii) to what *degree* their knowledge disagrees (e.g., wrt. an inconsistency degree, the KBs may still be *sufficiently consistent* s.t. the alignment may be “good enough” for collaborating). Here, our contributions are as follows:

- We present a novel approach for privacy-preserving inconsistency measurement; specifically, for two KBs K_A, K_B , we show how to compute two specific inconsistency measures for $K_A \cup K_B$ without revealing K_A, K_B (Section 3). To this aim, we show how private set intersections of sets of KB interpretations can be computed to measure various aspects of (in)consistency.
- We evaluate the developed methods by showing important privacy- and runtime complexity properties (Section 4).

We discuss preliminaries in Section 2 and conclude in Section 5. Proofs are shown in the appendix.

2 Preliminaries

2.1 Knowledge Bases, Inconsistency Measurement

In this work, we consider agents carrying internal knowledge in form of propositional logic knowledge bases. For this, let At be some fixed propositional signature, i. e., a (possibly infinite) set of propositions, and let $\mathcal{L}(\text{At})$ be the corresponding propositional language constructed using the connectives \wedge, \vee and \neg .

Definition 1. *A knowledge base K is a finite set of formulas $K \subset \mathcal{L}(\text{At})$.*

For a set of formulas X , we denote the set of contained propositions as $\text{At}(X)$. An interpretation ω on At is a function $\omega : \text{At} \rightarrow \{0, 1\}$ (where 0 stands for false and 1 stands for true). Let $\Omega(\text{At})$ denote the set of all interpretations for At . An interpretation ω *satisfies* (or is a *model* of) an atom $a \in \text{At}$, denoted by $\omega \models a$, iff $\omega(a) = 1$. The satisfaction relation \models is extended to formulas in the usual way. For $\Phi \subseteq \mathcal{L}(\text{At})$ we also define $\omega \models \Phi$ if and only if $\omega \models \phi$ for every $\phi \in \Phi$. For a set of formulas X , the set of models is $\text{Mod}(X) = \{\omega \in \Omega(\text{At}) \mid \omega \models X\}$. If $\text{Mod}(X) = \emptyset$ we write $X \models \perp$ and say that X is *inconsistent*.

An inconsistency *measure* \mathcal{I} is a function that assigns a non-negative numerical value to a knowledge base. The concrete behaviour of inconsistency measures is driven by rationality postulates. In this work, we assume inconsistency measures \mathcal{I} satisfy the basic property of *consistency* (for a KB K):

Consistency CO $\mathcal{I}(K) = 0$ iff $K \not\models \perp$

Numerous inconsistency measures have been proposed (see [12] for a survey). In this work, we consider the *drastic* inconsistency measure \mathcal{I}_d [6] and the *consistency* inconsistency measure \mathcal{I}_c [5], which we define below. In order to define

the contension measure we need some additional background on three-valued logic [9]. A three-valued interpretation is a function $\nu : \text{At} \rightarrow \{0, 1, \text{both}\}$, which assigns to every atom either 0, 1 or both, where 0 and 1 correspond to *false* and *true*, respectively, and both denotes a conflict. Assuming the *truth order* \prec_T with $0 \prec_T \text{both} \prec_T 1$, the function ν can be extended to arbitrary formulas as follows: $\nu(\alpha \wedge \beta) = \min_{\prec_T}(\nu(\alpha), \nu(\beta))$, $\nu(\alpha \vee \beta) = \max_{\prec_T}(\nu(\alpha), \nu(\beta))$, $\nu(\neg\alpha) = 1$ if $\nu(\alpha) = 0$, $\nu(\neg\alpha) = 0$ if $\nu(\alpha) = 1$, and $\nu(\neg\alpha) = \text{both}$ if $\nu(\alpha) = \text{both}$. We say an interpretation ν satisfies a formula α , denoted by $\nu \models^3 \alpha$, iff $\nu(\alpha) = 1$ or $\nu(\alpha) = \text{both}$. We are now ready to define the considered inconsistency measures.

Definition 2 (Considered Inconsistency Measures). *Given a knowledge base K , define $\mathcal{I}_d, \mathcal{I}_c$ via:*

$$\mathcal{I}_d(K) = \begin{cases} 1 & \text{if } K \models \perp \\ 0 & \text{otherwise} \end{cases} \quad \mathcal{I}_c(K) = \min\{|\nu^{-1}(\text{both})| \mid \nu \models^3 K\}$$

Example 2. Consider $K_1 = \{a; a \rightarrow b; \neg b \wedge \neg a; c\}$, then we have that $\mathcal{I}_d(K_1) = 1$ and $\mathcal{I}_c(K_1) = 2$. (for \mathcal{I}_c , note that the three-valued interpretation ν_1 with $\nu_1(a) = \text{both}$; $\nu_1(b) = \text{both}$; $\nu_1(c) = 1$ is the only three-valued model that assigns both to a minimal number of atoms).

In this work—for two knowledge bases K_A, K_B of parties A, B , respectively—we compute $\mathcal{I}_d(K_A \cup K_B)$ and an upper-bound for $\mathcal{I}_c(K_A \cup K_B)$, without A and B having to reveal the contents of their knowledge bases to each other. To allude to some of our results, this will be achieved by comparing the *models* for the individual knowledge bases (also in a privacy-preserving way). For example, we can exploit that $\text{Mod}(K_1) \cap \text{Mod}(K_2) = \emptyset$ iff $K_1 \cup K_2 \models \perp$, which allows to verify consistency without revealing formulas. Intuitively, the interpretations should also not be revealed, which we will show how to handle. In the following subsections, we discuss important notions and methods from a security perspective.

2.2 Cryptographic Techniques

In this work, we consider (asymmetric) encryption schemes, or cryptosystems, that can securely encode and decode messages with algorithmic techniques.

Definition 3 (Cryptosystem, [11]). *Let \mathcal{M} be a set of messages, called a message space, and let $\rho \in \mathbb{N}$ be a security parameter. Then, an encryption scheme is a tuple $(\mathbf{K}, \mathbf{E}, \mathbf{D})$, where*

- \mathbf{K} is a (key generation) function that takes the security parameter ρ and returns a key pair (k_e, k_d) for encryption/decryption, with $k_e \in \mathcal{K}_e, k_d \in \mathcal{K}_d$ (with $\mathcal{K}_e, \mathcal{K}_d$ being key spaces).
- \mathbf{E} is an (encryption) function $\mathbf{E} : \mathcal{K}_e \times \mathcal{M} \rightarrow \mathcal{C}$ that returns a ciphertext for a plaintext m , where \mathcal{C} is a ciphertext space.
- \mathbf{D} is a (decryption) function $\mathbf{D} : \mathcal{K}_d \times \mathcal{C} \rightarrow \mathcal{M}$ that takes a ciphertext and outputs a plaintext m , s.t. if $c = \mathbf{E}(k_e, m)$ then $\text{Probability}[\mathbf{D}(k_d, c) \neq m]$ is negligible, i.e., $\text{Probability}[\mathbf{D}(k_d, c) \neq m] \leq 2^{-\rho}$.

We consider encryption functions that are probabilistic, i.e., \mathbf{E} can return different ciphertexts even for two equal inputs (on the other hand, \mathbf{D} is deterministic) [11]. To clarify, given two plaintexts $m_1 = m_2 = 1$ and a key pair k_e, k_d produced by the encryption scheme, we have that $\text{Probability}[\mathbf{E}(k_e, m_1) = \mathbf{E}(k_e, m_2)] \leq 2^{-\rho}$, but $\mathbf{D}(k_d, \mathbf{E}(k_e, m_1)) = \mathbf{D}(k_d, \mathbf{E}(k_e, m_2))$. This is also referred to as the encryption scheme being IND-CPA secure (ciphertext indistinguishability under chosen plaintext attacks) [3]. We use $\mathbf{E}(k_e, m_1) \equiv \mathbf{E}(k_e, m_2)$ to denote that two ciphertexts carry “semantically” the same value, even though the ciphertexts are not identical.

For the technical development of our techniques, we assume two communicating parties, where both parties act via a *honest-but-curious* adversarial model [3], i.e., parties do not deviate from the protocol but may try to infer additional information from the data they obtain. We comment on the effects of a party taking on other adversarial models in Section 4.

2.3 Secure Multi-Party Computation

An important cryptographic field we build on is that of secure multi-party computation (SMPC) [3]. The goal of SMPC approaches is to allow multiple parties P_1, \dots, P_n to compute a function f over their (respective) inputs x_1, \dots, x_n , without revealing the inputs to the other parties. Any protocol performing this computation should satisfy the following properties:

Input Privacy (IP) Inputs should not be revealed during computation.

Correctness (Cor) The revealed output is the actual result of $f(x_1, \dots, x_n)$.

We use the term “privacy-preserving” to denote that a computation satisfies IP.

To develop privacy-preserving inconsistency measurement techniques, we will devise protocols that can compute (various aspects of) intersections of sets of knowledge base models while satisfying IP, Cor. This type of protocol is referred to as private set intersection (PSI) (cf. Section 2.5). PSI protocols build on so-called homomorphic encryption schemes, which we introduce next.

2.4 Homomorphic Encryption

Homomorphic encryption [13] is a cryptographic method that allows certain mathematical operations to be performed directly on encrypted data. Specifically, we consider *fully homomorphic* encryption schemes [4], which allow addition and multiplications of numerical values while remaining encrypted.

Definition 4 (Homomorphic encryption scheme, [11]). *Let \mathcal{M} be a message space, and let ρ be a security parameter. Then, a homomorphic encryption scheme is a quadruple $(\mathbf{K}, \mathbf{E}, \mathbf{D}, \circ)$ as follows:*

- $\mathbf{K}, \mathbf{E}, \mathbf{D}$ are key-generation-, encryption- and decryption functions as before.
- \circ is an operator for which it holds that for all messages $m_1, m_2 \in \mathcal{M}$: if $m_3 = m_1 \circ m_2$, and $c_1 = \mathbf{E}(k_e, m_1)$, and $c_2 = \mathbf{E}(k_e, m_2)$, then $\text{Probability}[\mathbf{D}(k_d, c_1 \circ c_2) \neq m_3]$ is negligible.

In other words, a homomorphic encryption scheme is an encryption scheme with the property that the operation \circ is correctly preserved when performing it on the encrypted ciphertexts themselves.

In the following, we assume encryption schemes that are fully homomorphic (i.e., where \circ can be either addition (+) or multiplication (\times)), s.t. for all $m_1, m_2 \in \mathcal{M}, k_e \in \mathcal{K}_e$ we have (cf. [13]):

$$\begin{aligned} \mathbf{E}(k_e, m_1) + \mathbf{E}(k_e, m_2) &\equiv \mathbf{E}(k_e, m_1 + m_2), & m_1 + \mathbf{E}(k_e, m_2) &\equiv \mathbf{E}(k_e, m_1 + m_2), \\ \mathbf{E}(k_e, m_1) \times \mathbf{E}(k_e, m_2) &\equiv \mathbf{E}(k_e, m_1 \times m_2), & m_1 \times \mathbf{E}(k_e, m_2) &\equiv \mathbf{E}(k_e, m_1 \times m_2). \end{aligned}$$

Example 3. Let a key pair k_e, k_d produced by a fully homomorphic encryption scheme. For $m = 5$ and $c = \mathbf{E}(k_e, m) + 2$, we have $\mathbf{D}(k_d, c) = 7$.

Remark 1. We make the standard assumption that the size of the ciphertexts remains polynomially bounded (cf. the property of circuit-privacy in [1]). This ensures that the ciphertext obtained by performing an operation \circ on two inputs is hard to distinguish from a ciphertext obtained by encrypting a plaintext m .

This extends to vectors via element-wise operations. A survey of fully homomorphic encryption schemes can be found in [13].

2.5 Private Set Intersection

PSI protocols [7] are subtypes of SMPC that allow to compute the intersection of two sets without revealing the rest of the sets. There are many applications for PSI, for example, finding (only) the common friends in the contact lists of two parties without disclosing the full contact lists to each other. As an example, consider the following baseline PSI protocol.

Example 4. Let two parties A, B each with a set containing exactly one integer, respectively, x and y . To compute PSI, A employs a fully homomorphic encryption scheme to generate a key pair k_e, k_d and sends $\mathbf{E}(k_e, x)$ to B . Then, B computes $c = r * (\mathbf{E}(k_e, x) - y)$ (which is a ciphertext, cf. Remark 1), where r is a random nonzero integer chosen by B . A now computes $\mathbf{D}(k_d, c)$: If the result is 0, x and y are identical, otherwise, $\{x\} \cup \{y\} = \emptyset$. In the latter case, A cannot infer any information about y (as r is chosen by B). In any case, B cannot infer anything about x (beyond the guarantees of the encryption scheme) as B only obtains ciphertext. The protocol can be performed symmetrically.

In this work, we consider versions of PSI protocols where only the size of the intersection is revealed. As we will show, inconsistency can then be characterized with various aspects of intersection sizes for sets of interpretations.

3 Approaches for Privacy-Preserving Inconsistency Measurement

For the remainder, we fix two parties A, B with respective knowledge bases K_A, K_B . Also, we fix a fully homomorphic encryption scheme $(\mathbf{K}, \mathbf{E}, \mathbf{D}, \circ)$ and

a corresponding key-pair k_e, k_d . The goal of this section is then two-fold: first, we develop an SMPC protocol allowing to compute $\mathcal{I}_d(K_A \cup K_B)$; then, to allow for a more gradual measure, we develop an SMPC protocol allowing to compute an upper-bound for $\mathcal{I}_c(K_A \cup K_B)$ (both protocol satisfying IP and Cor). For the remainder, we assume the KBs K_A, K_B on their own are consistent (and the task is to assess the consistency of $K_A \cup K_B$).

From IP, it is immediate that no formulas must be revealed for the computations. Instead, in our protocols, the parties will exchange their respective models; as discussed, consistency can then be verified by checking whether $\text{Mod}(K_A) \cap \text{Mod}(K_B) \neq \emptyset$. An important remark here is that the interpretations/models should also not be revealed in plain form. Otherwise, it would be possible to disjunctively write each interpretation as a conjunction of atoms, which would yield a formula in disjunctive normal form that is equivalent to the KB. Thus, we introduce PSI-based protocols that allow to *privately* compute the intersection of $\text{Mod}(K_A)$ and $\text{Mod}(K_B)$. For this, we need some further notation.

For any interpretation ω over At , we will encode ω as a bit sequence of 1s and 0s, which indicates the truth value of the atoms in alphabetical order.

Example 5. Let $\text{At} = \{a, b, c\}$; then we write $\omega_1 = 101$ to encode $\omega_1(a) = 1$, and $\omega_1(b) = 0$, and $\omega_1(c) = 1$.

This encoding will be used in various encryption processes. For example, we can directly encrypt interpretations by encrypting the encoding – in the example, $\mathbf{E}(k_e, 101)$. A further remark is that, also wrt. Axioms (1)-(4), this shorthand notation is useful for comparing interpretations via a bitwise comparison. For example, for two interpretations 11 and 10, a bitwise comparison 01 indicates that the first digit is identical and the second digit differs.

We are now ready to define a core protocol for comparing *two* interpretations. This core protocol will then later be used in two subsequent protocols (for $\mathcal{I}_d, \mathcal{I}_c$).

3.1 General Protocol for Privacy-Preserving Comparison of (Two) Interpretations

Assume two parties A and B who each have one interpretation (ω_A, ω_B) . For example, this interpretation could be derived from their knowledge bases under the closed-world assumption (if an atom is not entailed by the knowledge base, it is supposed to be false). Importantly, the two parties agree on a shared set of atoms At (needed to produce the encoding of the interpretations). We now want to verify if these interpretations are compatible. More precisely, Algorithm 1 specifies an SMPC protocol that takes as input two interpretations ω_A, ω_B and returns the number of atoms that they interpret differently. For the protocol, we recall the introduced encoding that allows to represent interpretations as binary numbers. For any binary number w , let $\text{len}(w)$ denote the number of digits of w , and w_i the i^{th} bit of w .

Algorithm 1 (Alg1) Compute Number of Differing Truth Assignments Given Interpretations A, B

Input: Shared set of atoms At , Interpretations ω_A, ω_B (over At , in shorthand notation)

Output: Number of truth assignments differing between ω_A and ω_B

- 1: A generates key pair (k_e, k_d)
 - 2: A generates a vector $v = \langle \omega_{A_1}, \dots, \omega_{A_{\text{len}(\omega_A)}} \rangle$
 - 3: A computes $v_{enc} = \langle \mathbf{E}(k_e, \omega_{A_1}), \dots, \mathbf{E}(k_e \omega_{A_{\text{len}(\omega_A)}}) \rangle$
 - 4: A sends v_{enc} to B
 - 5: B generates a vector $v_B = \langle \omega_{B_1}, \dots, \omega_{B_{\text{len}(\omega_B)}} \rangle$
 - 6: B computes $v_{A \oplus B} = v_{enc} - v_B$ {"XOR" operation; B cannot read result}
 - 7: B computes $n = \sum_{i=1}^{|v_{A \oplus B}|} (v_{A \oplus B}[i])^2$ {B cannot read result}
 - 8: B sends n to A
 - 9: A computes $n' = \mathbf{D}(k_d, n)$
 - 10: **return** n'
-

Example 6. Assume two parties A, B with interpretations $\omega_A = 110$ and $\omega_B = 101$, respectively.

- A generates a key pair.
- A generates the vector $v = \langle 1, 1, 0 \rangle$, resp. $v_{enc} = \langle \mathbf{E}(k_e, 1), \mathbf{E}(k_e, 1), \mathbf{E}(k_e, 0) \rangle$.
- B generates the vector $v_B = \langle 1, 0, 1 \rangle$.
- B computes $v_{A \oplus B} = v_{enc} - v_B = \langle \mathbf{E}(k_e, 1) - 1, \mathbf{E}(k_e, 1) - 0, \mathbf{E}(k_e, 0) - 1 \rangle$ (note that each position is still a ciphertext that B cannot read).
- B computes $n = \sum_{i=1}^{|v_{A \oplus B}|} (v_{A \oplus B}[i])^2$ (sum of absolute values; result is a ciphertext).
- A receives n and returns $n' = \mathbf{D}(k_d, n) = 2$

In result, A can infer that ω_A, ω_B differed in 2 assignments, but, importantly, cannot infer at which assignments the interpretations differ. The protocol can be performed symmetrically to produce the result for B .

Theorem 1. *Algorithm 1 satisfies IP, Cor.*

The presented core protocol allows to correctly compare the number of differing truth assignments for two interpretations, without revealing them. This core protocol will now be leveraged for computing further measures, specifically, by extending the protocol to take as input not only one interpretation, but two respective sets of interpretations/models by A and B .

3.2 Privacy-Preserving Computation of \mathcal{I}_d

We recall the definition of \mathcal{I}_d and parties A, B with knowledge bases K_A, K_B . To compute \mathcal{I}_d , we build on the fact that $\text{Mod}(K_A) \cap \text{Mod}(K_B) = \emptyset$ iff $K_A \cup K_B \models \perp$. To leverage Algorithm 1 for computation, we create two bit sequences as follows: First, A, B agree on a shared set of atoms At . Then, both parties independently

create a truth table showing satisfaction of their knowledge base, where the truth-assignments (rows) are in ascending binary order, ensuring equally ordered and exhaustive enumeration of all possible truth value combinations.

Example 7. Consider two agents A and B , each with their KBs K_A and K_B , respectively, with $K_A = \{a \wedge b\}$ and $K_B = \{\neg a\}$. Then, A and B construct the following truth tables:

$$\begin{array}{c}
 \begin{array}{c|c} a & b \\ \hline 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{array} \models K_A &
 \begin{array}{c|c} a & b \\ \hline 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{array} \models K_B \\
 A : & B :
 \end{array}$$

Both parties then take the last column, which is a bit-sequence. In the example, we get the two sequences $S_A = 0001$ and $S_B = 1100$. It is important to note that (as the row index for both tables correspond due to the ascending order), each index i over both sequences exactly encodes whether the i^{th} assignment of truth values satisfies K_A , resp., K_B . We then leverage these sequences as input for Algorithm 1 to verify consistency. For this, we define a slight variation of Algorithm 1, denoted Algorithm 1^{binary}, where we change Lines 6 and 7 as follows: i) 6: B computes $v_{A \oplus B} = 1 - (v_{enc} * v_B)$; ii) 7: B computes $n = \prod_{i=1}^{|v_{A \oplus B}|} (v_{A \oplus B}[i])^2$.

This has the following impact on the algorithm output: For line 6, if the two multiplicands differ or are both 0 (corresponding to inconsistency or non-satisfaction), the entry computed in line 6 is 1 (indicating a distance). Otherwise, it is 0 (indicating correspondence and satisfaction). In result, if at least one row satisfies both knowledge bases, the result of Algorithm 1^{binary} is simply 0 (cf. line 7). If all rows differ, the returned value in this way is exactly 1, hiding the number of rows. This binary version of Algorithm 1 can be leveraged as follows.

Algorithm 2 (Alg2) Compute $\mathcal{I}_d(K_A \cup K_B)$

Input: Shared set of atoms At , knowledge bases K_A, K_B

Output: $\mathcal{I}_d(K_A \cup K_B)$

- 1: A, B create respective truth tables in ascending binary order (over At).
 - 2: A, B obtain (via the last column) their private sequences S_A, S_B .
 - 3: A, B compute $d = \text{Algorithm 1}^{\text{binary}}$ wrt. At, S_A, S_B
 - 4: **return** d
-

Example 8. We recall the KBs and truth tables from Example 7, with the two sequences $S_A = 0001, S_B = 1100$ (this relates to lines 1-2 of Alg2). Then, Algorithm 1^{binary} is computed wrt. S_A, S_B and stored as d . Regarding that Algorithm, recall that if (the assignments) of at least row match and satisfy both knowledge bases, $d = 0$, and $d = 1$ otherwise. In the example, indices 1,2,4 do not match, and, while the third index (i.e., the interpretation $\omega(a) = 1; \omega(b) = 0$) matches, this interpretation is not a model. In turn, the output is 1.

Theorem 2. *Algorithm 2 satisfies IP, Cor.*

3.3 Privacy-Preserving Approximation of \mathcal{I}_c

We continue with a gradual measure based on \mathcal{I}_c . For this, recall parties A, B with K_A, K_B . Then, consider Alg3, which works over *models* of K_A and K_B :

Algorithm 3 (Alg3) Compute Smallest Distinct Number of Mismatching Assignments for Any Pair in Two Sets of Models

Input: Shared set of atoms At , two sets of models: $\text{Mod}(K_A), \text{Mod}(K_B)$
Output: Smallest number of different assignments over all combinations of models

- 1: A initializes an empty set $S = \{\}$
- 2: **for all** element $A \in \text{Mod}(K_A)$ **do**
- 3: **for all** element $B \in \text{Mod}(K_B)$ **do**
- 4: Both A and B perform **Algorithm 1** wrt. At , element A , element B .
- 5: A stores the result in S
- 6: **end for**
- 7: **end for**
- 8: **return** $\text{MIN}(S)$

Example 9. Assume two parties A, B (with K_A, K_B) and $\text{Mod}(K_A) = \{111, 110\}$, $\text{Mod}(K_B) = \{100, 101\}$.

- Alg. 3 will iterate over all combinations of models and yields: $\text{Alg1}(111, 100) = 2$; $\text{Alg1}(111, 101) = 1$; $\text{Alg1}(110, 100) = 1$; $\text{Alg1}(110, 101) = 2$.
- The returned result is $\text{MIN}(\{1, 2\}) = 1$.

First, observe that the result of Example 9 can be interpreted s.t. no combination of models agree ($\text{Mod}(K_A \cup K_B) = \emptyset$). We now show how the result of Algorithm 3 can be used to approximate the contension measure. The idea is that we use the results of Algorithm 3 to derive a set of *three-valued* interpretations $\text{Mod}_3(K_A \cup K_B)$ (to plug into \mathcal{I}_c). For this, the following will be useful.

Lemma 1. *Let F be a formula and let I be a three-valued interpretation that satisfies F . Let J be a three-valued interpretation obtained from I by changing the interpretation of a single atom to both. Then $I(F) = J(F)$ or $J(F) = \text{both}$.*

Corollary 1. *Let I be a two-valued model of K and let J be a three-valued interpretation obtained from I by changing the interpretation of atoms to both. Then J is a three-valued model of K .*

For two KBs K_1, K_2 , we now use this result to define a function that can transform the two-valued interpretations $\text{Mod}(K_1), \text{Mod}(K_2)$ into a set of three-valued interpretations M_3 s.t. for every $m \in M_3$: $m \models_3 K_1 \cup K_2$.

Definition 5. *Let two knowledge bases K_1, K_2 and let two interpretations i_1, i_2 over At s.t. $i_1 \models K_1, i_2 \models K_2$. Then, define a three-valued interpretation via the*

function $f(i_1, i_2)$, where

$$f(i_1, i_2)(j) = \begin{cases} i_1(j), & \text{if } i_1(j) = i_2(j) \\ \text{both}, & \text{otherwise} \end{cases}$$

Corollary 1 implies that a three valued interpretation obtained via f is a three-valued model for $K_1 \cup K_2$. With a slight abuse of notation, we let:

$$f(\text{Mod}(K_1), \text{Mod}(K_2)) = \{f(i_1, i_2) \mid i_1 \in \text{Mod}(K_1), i_2 \in \text{Mod}(K_2)\}.$$

We now show the relationship of Algorithm 3 to \mathcal{I}_c .

Proposition 1. *Let x be the inconsistency value computed by Algorithm 3. Then $x \geq \mathcal{I}_c(K_A \cup K_B)$.*

However, Algorithm 3 can overestimate the inconsistency value.

Example 10. Let $K_A = \{a, a \rightarrow b_1 \wedge b_2\}$ and $K_B = \{a, a \rightarrow \neg b_1 \wedge \neg b_2\}$. Then $\text{Mod}(K_A) = \{111\}$ and $\text{Mod}(K_B) = \{100\}$. Hence, $f(\text{Mod}(K_A), \text{Mod}(K_B)) = \{1\text{bothboth}\}$ and Algorithm 3 will return 2. However, both00, both01, both10, both11 are also three-valued models of $K_A \cup K_B$. Therefore \mathcal{I}_c is 1.

While Algorithm 3 cannot compute the contention inconsistency value exactly, it will never underestimate the inconsistency (it gives an upper bound on the inconsistency value). Importantly, it will also never report a positive inconsistency value when the knowledge bases are, in fact, mutually consistent.

Proposition 2. *If $K_A \cup K_B$ is consistent, then Algorithm 3 will return 0.*

Let us note that Alg. 3 satisfies our privacy guarantee.

Theorem 3. *Algorithm 3 satisfies IP.*

While Algorithm 3 satisfies IP, it reveals more than the output⁴: Clearly, a) B learns the number of models provided by A , and b) A learns all distinct differences in truth assignments over all model combinations. We therefore show a slight variation (Alg4) which allows to counteract this.

In line 1, A creates a multiset with models of K_A of size $2^{|\text{At}|}$ (possibly containing duplicates). This is a padding that ensures A does not reveal the number of models. Lines 2-8 are analogous to Alg3, using the padded multiset. Lines 9-12 are an encrypted computation by B . L is a list from 0 to $|\text{At}|$, the possible range for \mathcal{I}_c . Then, we compute the distance between every L_i and all results in S : $i - d = 0$ must hold for at least one $d \in S$. We are only interested in the minimum of $0, \dots, |\text{At}|$ that is a match (lines 13-18), hence we obscure all left of the minimum by means of prime encryption and set all right of the minimum to zero (numbers that are not zero are “meta-encrypted” and stay encrypted even after A decrypts the result). The smallest distinct number of mismatching assignments is the first index in L of an element that is 0. This solves the problems with Con exhibited in Alg3: i) B cannot know the number of A ’s models as A sends a padded multiset, ii) A does not learn all distinct differences in truth assignments over all combinations.

⁴ This is also referred to as the SMPC properties of confidentiality (Con).

Algorithm 4 (Alg4) Compute Smallest Distinct Number of Mismatching Assignments for Any Pair in Two Sets of Models, Satisfying Con

Input: Shared set of atoms At , two sets of models: $\text{Mod}(K_A)$, $\text{Mod}(K_B)$

Output: Smallest number of different assignments over all combinations of models

- 1: A initializes $\text{Mod}(K_A)'$ as multiset of length $2^{|\text{At}|}$, containing all and only elements of $\text{Mod}(K_A)$
- 2: B initializes an empty list $S = \langle \rangle$
- 3: **for all** element $A \in \text{Mod}(K_A)'$ **do**
- 4: **for all** element $B \in \text{Mod}(K_B)$ **do**
- 5: Both A and B perform **Algorithm 1** wrt. At , element A , element B .
- 6: B stores the (encrypted) result in S
- 7: **end for**
- 8: **end for**
- 9: B initializes an empty list $L = \langle \rangle$
- 10: **for all** $i \in \{0, \dots, |\text{At}|\}$ **do**
- 11: B computes $L_i \leftarrow \prod_{d \in S} (i - d)$
- 12: **end for**
- 13: **for all** $i \in \{0, \dots, |\text{At}|\}$ **do**
- 14: B computes $p \leftarrow$ random prime number
- 15: B computes $L_i \leftarrow (\prod_{0 \dots i} L_i)^{p-1}$
- 16: **end for**
- 17: A decrypts L
- 18: **return** Index of the first element in L that is 0

Example 11. Recall Example 9 with $\text{Mod}(K_A) = \{111, 110\}$, $\text{Mod}(K_B) = \{100, 101\}$. First, A will create a padded list of size $2^{|\text{At}|}$, here: $\text{Mod}(K_A)' = \{111, 110, 111, 110, 111, 110, 111, 110\}$. A and B perform lines 3-8 which yields a list of (encrypted) differences: Alg1 is performed for all (model, model)-tuples in $\text{Mod}(K_A)' \times \text{Mod}(K_B)$, yielding the (encrypted) list $\{\mathbf{E}(k_e, 2), \mathbf{E}(k_e, 1), \dots\}$. B then checks which of the potential distances in $\{0, 1, 2, 3\}$ exist in this list by executing lines 9-12, yielding the list $\langle 2, 0, 0, 2 \rangle$ (again, encrypted). The multiplication of every element in the list with its predecessors and subsequent prime encryption (by B) in lines 13-14 results in the list $\langle \text{enc}_p, 0, 0, 0 \rangle$, where enc_p is the prime-encrypted 2. Finally, A decrypts the list. However, enc_p is useless (as it was meta-encrypted). A can only infer from $\langle \text{enc}_p, 0, 0, 0 \rangle$ that $\mathcal{I}_c = 1$ (index of the first 0).

While Alg4 comes with improvement wrt. Con, A has to create a padded multiset of exponential size. In the following we discuss this trade-off.

4 Discussion

We start by showing the upper and lower bounds of runtime-complexity in Table 1 (proofs in appendix).

While Alg3 violates Con, it can approach polynomial scaling in best-cases. Alg4 retains its exponential component due to padding, trading off complexity and privacy requirements, depending on the use-case. Alg2 scales exponentially.

	\mathcal{O}	Ω
Algorithm 1	$\mathcal{O}(k + \text{At})$	$\Omega(k + \text{At})$
Algorithm 2	$\mathcal{O}(k + 2^{ \text{At} })$	$\Omega(k + 2^{ \text{At} })$
Algorithm 3	$\mathcal{O}(k + 2^{2^{ \text{At} }} * \text{At})$	$\Omega(k + \text{At})$
Algorithm 4	$\mathcal{O}(k + 2^{2^{ \text{At} }} * \text{At})$	$\Omega(k + 2^{ \text{At} } * \text{At})$

Table 1: Runtime-complexity (upper (\mathcal{O})/lower (Ω)) of the developed algorithms wrt. At ; k = cost of key generation.

For the discussion of algorithms we have considered an *honest-but-curious* adversarial model. Threats in this setting are bounded by the compliance with IP. For adversarial models such as *malicious adversary* (participants may deviate from the protocol), the guarantees given by IP also hold. However, intuitively, such a threat model can affect the correctness of the results: A malicious adversary can deliberately provide altered models of his/her own knowledge base, e.g., flipping all bits (note that the ability of the adversary to manipulate ciphertexts is mitigated by IND-CPA security). Likewise, the adversary could provide models even if the KB is inconsistent. While preventing the adversary to provide fake models cannot be mitigated, various methods exist to prove the consistency of the own KB (without revealing it) via zero-knowledge proofs [2], which can be put before our algorithms if needed.

One should also be aware of risks by repeated queries. Given no restrictions wrt. the number of queries sent, A could reveal information about B 's KB by altering the input for different queries. For this risk, it is important to consider what we actually reveal. Accordingly, we observe the worst-case probabilities with which one agent can successfully guess a model in another agents' KB.

Proposition 3. *A can correctly guess a model in K_B with a probability of at least $\frac{1}{|\text{Mod}(K_A)|}$ if K_A and K_B are consistent; if K_A and K_B are inconsistent, A can correctly guess with a probability of at least $\frac{1}{|\Omega(\text{At}) \setminus \text{Mod}_{<\text{Alg4}}|}$, where $\text{Mod}_{<\text{Alg4}} := \{m | m \in \Omega(\text{At}), \text{Alg4}(\text{Mod}(K_a), \{m\}) < \text{Alg4}(\text{Mod}(K_A), \text{Mod}(K_B))\}$.*

For both cases, A may straightforwardly reveal a formula equivalent to B 's KB by measuring $g(\{m\}, \text{Mod}(K_B))$ for all $m \in \Omega(\text{At})$, where $g \in \{\text{Alg2}, \text{Alg3}, \text{Alg4}\}$.

5 Conclusion

We have introduced novel methods for privacy-preserving inconsistency measurement. By leveraging SMPC and homomorphic encryption, the proposed algorithms enable agents to collaboratively evaluate the consistency of their KBs without revealing sensitive information. While the approach successfully implements input privacy, it also highlights trade-offs in runtime complexity and potential risks in adversarial settings. Overall, the framework advances the state of the art by enabling cooperative inconsistency measurement in privacy-critical settings. Future work can, for example, focus on methods for privately computing interpolants/common knowledge or disagreement between > 2 parties [8,10].

References

1. Armknecht, F., Boyd, C., Carr, C., Gjøsteen, K., Jäschke, A., Reuter, C.A., Strand, M.: A guide to fully homomorphic encryption. Cryptology ePrint Archive (2015)
2. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Proceedings of the 2012 ACM conference on Computer and communications security. pp. 784–796 (2012)
3. Cramer, R., Damgård, I.B., et al.: Secure multiparty computation. Cambridge University Press (2015)
4. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the forty-first annual ACM symposium on Theory of computing. pp. 169–178 (2009)
5. Grant, J., Hunter, A.: Measuring consistency gain and information loss in stepwise inconsistency resolution. In: European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty. pp. 362–373. Springer (2011)
6. Hunter, A., Konieczny, S., et al.: Measuring inconsistency through minimal inconsistent sets. KR **8**(358-366), 42 (2008)
7. Morales, D., Agudo, I., Lopez, J.: Private set intersection: A systematic literature review. Computer Science Review **49**, 100567 (2023)
8. Potyka, N.: Measuring disagreement among knowledge bases. In: International Conference on Scalable Uncertainty Management. pp. 212–227. Springer (2018)
9. Priest, G.: The logic of paradox. Journal of Philosophical logic pp. 219–241 (1979)
10. Ribeiro, J.S., Sofronie-Stokkermans, V., Thimm, M.: Measuring disagreement with interpolants. In: Scalable Uncertainty Management: 14th International Conference, SUM 2020, Bozen-Bolzano, Italy, September 23–25, 2020, Proceedings 14. pp. 84–97. Springer (2020)
11. Sen, J.: Homomorphic encryption-theory and application. Theory and practice of cryptography and network security protocols and technologies **31** (2013)
12. Thimm, M.: Inconsistency measurement. In: Scalable Uncertainty Management: 13th International Conference, SUM 2019, Compiègne, France, December 16–18, 2019, Proceedings 13. pp. 9–23. Springer (2019)
13. Yi, X., Paulet, R., Bertino, E., Yi, X., Paulet, R., Bertino, E.: Homomorphic encryption. Springer (2014)

Appendix: Proofs for Technical Results

Theorem 1. *Algorithm 1 satisfies IP, Cor.*

Proof. The protocol adheres to IP as no party learns anything beyond the Hamming distance, where the Hamming distance itself is the agreed-upon output (in particular, B works on ciphertexts only in line 6 and 7). The potential to infer complete input information in edge cases (e.g., output = 0 or $|\text{At}|$) is intrinsic to the meaning of the output and not an additional leakage under typical definitions. It is simply a characteristic of the output itself. For Cor, proceed by invariance: at each step, the algorithm accurately tracks the number of differing truth assignments between ω_A and ω_B . Initially, the vectors v and v_B represent the truth assignments of ω_A and ω_B , respectively. The XOR operation in $v_{A \oplus B}$ ensures that each entry reflects whether the corresponding truth assignments differ (1) or

are identical (0), even under encryption. The summation step correctly accumulates the total number of differing assignments - squaring each $v_{A \oplus B}[i]$ does not change its value since $v_{A \oplus B}[i] \in \{0, 1\}$. Finally, decryption reveals the correct count without altering the result. This invariant holds throughout the algorithm, ensuring its correctness.

Theorem 2. *Algorithm 2 satisfies IP, Cor.*

Proof. The protocol adheres to IP as no party learns anything beyond a binary assessment of consistency, where the assessment itself is the agreed-upon output. Cor is established by maintaining an invariant: At each step, the algorithm correctly tracks whether there exists at least one model that satisfies both knowledge bases. In line one, both parties create sequences encoding satisfaction, where each index is in the same order and corresponds to the same truth value assignment(s). In line 6, for every such index, Algorithm 1^{binary} returns 0, if the values are both 1, and 1, otherwise. Then via multiplication, if there exists at least one 0, the product is also 0. This ensures that the algorithm will return 0 if there is at least one interpretation satisfying both K_A, K_B , and 1, otherwise.

Lemma 1. *Let F be a formula and let I be a three-valued interpretation that satisfies F . Let J be a three-valued interpretation obtained from I by changing the interpretation of a single atom to both. Then $I(F) = J(F)$ or $J(F) = \text{both}$.*

Proof. We prove the claim by structural induction. We can assume w.l.o.g. that we change an atom that is contained in the formula F because changing the truth value of another atom cannot affect the interpretation by truth-functionality of the logical connectives.

For the base case, assume that F is an atom. If we change the interpretation of F to both, we have $J(F) = \text{both}$.

Since all formulas can be expressed using only \neg and \wedge , it is sufficient to consider these cases for the induction step.

Consider $F = \neg G$. If $I(\neg G) = 0$, then $I(G) = 1$ and by the induction assumption, we have $J(G) = 1$ or $J(G) = \text{both}$. Thus, $J(\neg G) = 0 = I(\neg G)$ or $J(\neg G) = \text{both}$. If $I(\neg G) = 1$, then $I(G) = 0$ and by the induction assumption, we have $J(G) = 0$ or $J(G) = \text{both}$. Thus, $J(\neg G) = 1 = I(\neg G)$ or $J(\neg G) = \text{both}$. If $I(\neg G) = \text{both}$, then $I(G) = \text{both}$ and by the induction assumption, we have $J(G) = \text{both} = I(G)$.

Consider $F = G_1 \wedge G_2$. If $I(G_1 \wedge G_2) = 0$, then $I(G_1) = 0$ or $I(G_2) = 0$. Assume w.l.o.g. that $I(G_1) = 0$ (the case $I(G_2) = 0$ is analogous). By the induction assumption, $J(G_1) = 0$ or $J(G_1) = \text{both}$. Hence, $J(G_1 \wedge G_2) = 0$ or $J(G_1 \wedge G_2) = \text{both}$. If $I(G_1 \wedge G_2) = 1$, then $I(G_1) = 1$ and $I(G_2) = 1$. By the induction assumption, ($J(G_1) = 1$ or $J(G_1) = \text{both}$) and ($J(G_2) = 1$ or $J(G_2) = \text{both}$). Hence, $J(G_1 \wedge G_2) = 1$ or $J(G_1 \wedge G_2) = \text{both}$. If $I(G_1 \wedge G_2) = \text{both}$, then $I(G_1) = \text{both}$ or $I(G_2) = \text{both}$. Assume w.l.o.g. that $I(G_1) = \text{both}$ (the case $I(G_2) = \text{both}$ is analogous). By the induction assumption, $J(G_1) = \text{both}$. Hence, $J(G_1 \wedge G_2) = \text{both}$.

Corollary 1. *Let I be a two-valued model of K and let J be a three-valued interpretation obtained from I by changing the interpretation of atoms to **both**. Then J is a three-valued model of K .*

Proof. For all formulas $F \in K$, we have $I(F) = 1$ by assumption. Note that two-valued interpretations are a special case of three-valued interpretations. When J changes the truth value of k atoms to **both**, it can be seen as a sequence J_0, J_1, \dots, J_k of three-valued models, $J_0 = I, J_k = J$ and J_i is obtained from $J_{i-1}, i = 1, \dots, k$ by changing the interpretation of a single atom to **both**. Hence, Lemma 1 guarantees that $J_i(F) = 1$ or $J_i(F) = \text{both}$ for all $i = 1, \dots, k$. Hence, J satisfies K .

Proposition 1. *Let x be the inconsistency value computed by Algorithm 3. Then $x \geq \mathcal{I}_c(K_A \cup K_B)$.*

Proof. Corollary 1 guarantees that the 3-valued interpretations computed by Algorithm 3 are models of three-valued models of $K_A \cup K_B$. Since x is the minimal number of **both** found across these models, and $\mathcal{I}_c(K_A \cup K_B)$ is the minimal number across all models, we must have $x \geq \mathcal{I}_c(K_A \cup K_B)$.

Proposition 2. *If $K_A \cup K_B$ is consistent, then Algorithm 3 will return 0.*

Proof. If $K_A \cup K_B$ is consistent, there must be an $i \in \text{Mod}(K_A \cup K_B)$. Hence, Algorithm 3 will compute $f(i, i) = i$ and therefore return 0.

Theorem 3. *Algorithm 3 satisfies IP.*

Proof. The protocol adheres to IP as no party learns the concrete models. A only learns the number of differing truth assignments but cannot map this to specific models.

Proposition 3. *The runtime complexity of Alg1 is $\mathcal{O}(k + |\text{At}|)$, where k is the asymptotic cost of generating the key pair.*

Proof. We proceed by line. Line 1 has cost k (see above). Line 2 and 3 perform constant operations over all $|\text{At}|$ positions of the interpretation ($\mathcal{O}(|\text{At}|)$). Line 4 is only part of the communication. Line 5 is analogous to line 2. Subtracting two vectors of length $|\text{At}|$ (line 6) is $\mathcal{O}(|\text{At}|)$. Note that $v_{A \oplus B}$ is also of size $|\text{At}|$. Line 7 performs a constant operation over $|\text{At}|$ positions. Line 8 is analogous to 4. Line 9 is constant. Thus we have $\mathcal{O}(k + 5 * |\text{At}| + 1) = \mathcal{O}(k + |\text{At}|)$.

Proposition 4. *The lower-bound runtime complexity of Alg1 is $\Omega(k + |\text{At}|)$, where k is the asymptotic cost of generating the key pair.*

Proof. Analogous to \mathcal{O} .

Proposition 5. *The runtime complexity of Alg2 is $\mathcal{O}(k + 2^{|\text{At}|})$, where k is the asymptotic cost of generating the key pair.*

Proof. Straightforward from Alg1 (Note that both parties construct a bit-sequence of length n , where n is $2^{|\text{At}|}$).

Proposition 6. *The lower-bound runtime complexity of Alg2 is $\Omega(k + 2^{|\text{At}|})$, where k is the asymptotic cost of generating the key pair.*

Proof. Analogous to \mathcal{O} .

Proposition 7. *Let A have s models and B have t models. The runtime complexity of Alg3 is $\mathcal{O}(k + s * t * |\text{At}|)$ (assuming the key is only generated once), where k is the asymptotic cost of generating the key pair. As both parties could have up to $2^{|\text{At}|}$ models, this relates to $\mathcal{O}(k + 2^{2^{|\text{At}|}} * |\text{At}|)$.*

Proof. The outer loop iterates over all s models in $\text{Mod}(K_A)$. The inner loop iterates over all t models in $\text{Mod}(K_B)$. For each pair of models (each of length $|\text{At}|$), Alg1 is called, which has a complexity of $\mathcal{O}(k + |\text{At}|)$.

Proposition 8. *Let A have s models and B have t models. The lower-bound runtime complexity of Alg3 is $\Omega(k + |\text{At}|)$, where k is the asymptotic cost of generating the key pair.*

Proof. In general the costs are $(k + s * t * |\text{At}|)$ (assuming the key is only generated once), where k is the cost of generating the key pair. In the best case, both parties have 1 model each (recall the KBs are consistent per assumption). This relates to $\Omega(k + 1 * 1 * |\text{At}|)$.

Proposition 9. *Let A have s models and B have t models. The runtime complexity of Alg4 is $\mathcal{O}(k + 2^{2^{|\text{At}|}} * |\text{At}|)$ (assuming the key is only generated once), where k is the asymptotic cost of generating the key pair.*

Proof. Lines 1-8 are analogous to Alg3 ($\mathcal{O}(k + 2^{2^{|\text{At}|}} * |\text{At}|)$). Note both parties could have up to $2^{|\text{At}|}$ models, so S can have a size of $2^{|\text{At}|}$. Line 9 is constant. Line 10 calls $|\text{At}|$ times an operation that requires $2^{|\text{At}|}$ subtractions. Lines 13-16 run $|\text{At}|$ times. Line 17 is constant. So we have $\mathcal{O}(k + 2^{2^{|\text{At}|}} * |\text{At}| + 1 + |\text{At}| * 2^{|\text{At}|} + |\text{At}|) = \mathcal{O}(k + 2^{2^{|\text{At}|}} * |\text{At}|)$.

Proposition 10. *Let A have s models and B have t models. The lower-bound runtime complexity of Alg4 is $\Omega(k + 2^{|\text{At}|} * |\text{At}|)$, where k is the asymptotic cost of generating the key pair.*

Proof. A pads to $2^{|\text{At}|}$ models, so even if B has 1 model only, line 5 is called $2^{|\text{At}|}$ times, where line 5's cost is $|\text{At}|$. Lines 9-17 are analogous to \mathcal{O} .