The question 'do you know the secrets that I know?' is a tricky one. We explore what a protocol must do in order to provide an answer to such questions. We distinguish the 1-to-many case and the many-to-many case, and survey protocols which solve these cases. There are no protocols in the literature yet which solve these cases where the domain of possible secrets is huge, except for the protocols (T-1 and T-2) we will present in the next chapters.

Chapter 8

Knowledge Authentication

We will introduce the objective of the material presented in this chapter by a rather innocent real-world situation which has actually occurred:

> Geertje and Wouter are two friends and colleagues. Geertje has told Wouter in private that she is expecting a baby.[1] Just a few days later, Wouter meets the secretary at the coffee corner. The secretary looks expectantly to Wouter. Wouter would like to gossip with the secretary about Geertje's pregnancy. But Wouter has also promised Geertje not to disclose the secret of her pregnancy. If Wouter wants to keep his promise, he can only start gossiping about Geertje's pregnancy if he can be certain that the secretary already knew the secret. Wouter cannot simply ask the secretary whether she knew, because such a question would disclose the secret.
>
> Similarly, the secretary might know the secret, and could also have promised not to disclose the secret. In the case that both the secretary and Wouter know the secret, they are allowed to gossip.[2]

Is there a strategy for the secretary and Wouter that enables them to mutually establish whether they know of Geertje's pregnancy without disclosing this secret? The answer is yes. We will call protocols that solve this type of problem protocols for *knowledge authentication*: authentication based on the *knowledge* of an actor — instead of based on the identity or role of the actor.

Protocols for *knowledge authentication* can be used to grant somebody access to confidential information based on his or her knowledge, instead of (only)

[1] The baby was born on May 24, 2004. Her name is Marloes and she is really cute.

[2] Of course, they should make sure not to be overheard, and the coffee corner is probably not the best location for not being overheard.

based on his or her identity and role.[3] In the above example, successful authentication grants access to quotes and sentences which should be described as 'gossip'.

In this chapter, we will precisely define the the fundamental problem that is exemplified in the above story. We will survey and categorize what solutions for this problem exist in the literature. In Chapters 9 and 10, we will present our solutions to this problem, which are more general and more efficient than all existing solutions.

8.1 Application Areas of Gossip

The desire to gossip may be an interesting occasion to devise protocols and touch upon fundamental research issues, but the application areas of the protocols defined in this chapter reach further than the social talk at the coffee corner. We will present two application areas where 'cautious gossip' has valuable applications.

8.1.1 Police Investigations

The situation that initiated the design of protocols for *knowledge authentication* has been arguably a more important than coffee corner gossip:

> When a research team of the police performs an investigation on some crime, they register the people who are victims, the people who are witnesses, and the people who are suspects. If two independent crimes are investigated, it may be the case that somebody is suspect of two independent criminal acts. If the research teams of the two crimes do not communicate, they can easily harm one another's research. For example, one research team may be shadowing the suspect, and the other team may want to arrest the suspect. If the research teams do not communicate, it can be expected that something will go wrong in at least one of the investigations.
>
> How can a research team know that some other research team is investigating the same person?

One solution for this police problem could be that the police has an organization-wide notice board on which all victims, witnesses and suspects are listed. The current solution in the Dutch police[4] is not very much unlike this one: a police officer can enter a name in the computer, and will get a list of research teams investigating the person.

[3] This should not be confused with protocols for *private authentication* [AF04], which are protocols where the identity of a principal is only proven to a restricted set of possible communication partners.

[4] See Section 1.5 for a detailed description of the current solution in the Dutch police.

The Dutch police has thousands of criminal investigators. All of these investigators have made an oath which morally binds them to righteous behavior. The sheer number of police investigators entails that one can be sure that at least some of them will be corrupt and malicious. Thus, an organization-wide notice board that can only be consulted by officers who have made an oath is not a good solution; just a few corrupt officers who leak the information on the notice board can be enough to help some criminals escape the fate they deserve by law.[5]

It is justified to say that the organization-wide notice board is not the source of the problem, but that the few corrupt police officers are the source of the problem. If one is interested in addressing the problem of corrupt police officers, one should of course always tackle the problem at its root: hunt down and eliminate corrupt officers. It would not be realistic to believe that hunting down corrupt officers will be a 100% effective. Therefore, some measures have to be taken to limit the negative impact that corrupt police officers can have.

A first step in limiting the negative impact that corrupt police officers can have, is to prevent 'frivolous queries', that is, to prevent queries for which there does not exist a *need to know*. Of course, the *need to know* is something which is often hard to operationalize precisely. But even if it is only operationalized in a rough, simplistic way, such that only obvious violations are detected, it already limits the impact of corrupt police officers.

A simple operationalization of *need to know* which dramatically limits the frivolous queries of the thousands of researchers: only queries are allowed on names which occur in the electronic dossier the researcher is working on.[6]

There are also police officers which operate the organization-wide notice board, and if the information is stored unencrypted on the notice board[7] these operators will still be able to perform frivolous queries at will.

As long as the software that is running the notice board needs access to unencrypted data, the operators will have a means to access the unencrypted data. However, when the software that operates the notice board does not need access to unencrypted data to match electronic dossiers, it is possible to prevent frivolous access by malicious operators. When protocols for *knowledge authentication* are applied, notice boards do not need unencrypted data.

8.1.2 The Passenger Name Record

The best known application area where protocols for *knowledge authentication* can help is the airline passenger data (the so-called *passenger name record* or PNR). In 2003, in response to the events on 9/11, the United States of America

[5] There have been major leaks in the Dutch police organization, as Dutch top-criminals like Mink Kok have possession of various classified police documents. The police has a hard time identifying the corrupt police officers (various newspaper media, 2006).

[6] Such an operationalization has to be enforced by the software which is used to manage the electronic dossier.

[7] Or equivalently: it is stored encrypted, but the operators have access to the decryption key.

mandated that all airline carriers release the PNR to the Department of Homeland Security (DHS) for all flights from, to, and over US territory:

> The DHS has a 'terrorist list' of people they do not wish close to US territory. When an airplane wants to enter US airspace and it carries one or more people who are on the 'terrorist list', it is refused access.[8] Understandably, the DHS does not want to disclose its terrorist list; such a disclosure would give an unnecessary advantage to terrorists. Al Qaeda would precisely know which of its combattants would be granted access to the US, and which not.
>
> The airline carriers, on the other hand, are not automatically inclined to release the PNR to the DHS. The information has been collected for commercial purposes, and not for security purposes. Release of the PNR would result in infringement on the privacy of innocent citizens. The European Data Protection Directive forbids the release of this information. This resulted in a circus of lawsuits and negotiations.[9]
>
> Can the airline carriers and the DHS compare their lists of passengers and suspected terrorists without mutually disclosing their lists?

This problem is more intricate than the problem of the police investigation information, because the airline carriers and the DHS are not subsidiaries of one larger organization. As such, it will be hard — if not impossible — to find a *trusted third party* (TTP) to compare their lists. Thus, a 'notice board solution' as with the police investigation information is impossible.

Using protocols for *knowledge authentication*, it is possible to determine the intersection of two lists, without disclosing the lists themselves[10], without the need for a TTP.

When we look at the *need to know* of the DHS, we can observe that it is in fact very limited. What the DHS needs to know is *whether* there is a suspected terrorist on board of an airplane. Strictly taken, the DHS does not even need to know *which* terrorist is on board. Importantly, the DHS does *not* need to know the identities of the passengers that are *not* suspected terrorists.

Thus, protocols for *knowledge authentication* can protect the privacy of innocent citizens who fly from, to, or over the US, while the DHS can still perform its task.[11]

[8] Remarkably, when a airplane is actually refused access, there is no procedure for a concerted effort to arrest the suspected terrorist. After the airplane lands outside of the US, the suspected terrorist is free to travel elsewhere.

[9] For an extensive treatment of how the European Union and the US settled their dispute on the PNR, consult [Hei05].

[10] Thus, the items which are on both lists are mututally disclosed, but the items which are only on one of the lists are kept secret.

[11] Using protocols for *knowledge authentication* the identity of suspected terrorists which are on board of an airplane *is* disclosed to the airline carrier, and in this sense the terrorist list is not kept secret. In the current situation in which the full passenger list is disclosed to the DHS, the

8.2 Comparing Information Without Leaking It and Reference

Protocols for *knowledge authentication* are for comparing secrets, without disclosing the secrets. We need to be more precise on what we consider to be 'secret', and what we mean by 'comparing without disclosing'. We will make this more precise in this section.

What constitutes a 'secret' is relatively simple. A secret of player Q is a bit string, generated by player Q, which player Q is not willing to disclose to others. Whether the bit string can be generated by virtually every other agent does not alter it being a secret of agent Q. Thus, Q may consider 'Stalin sent millions of people to Siberia' to be a secret, while in fact many people know this. Moreover, whether the bit string corresponds to something which is true in the 'outside world' is irrelevant: for example, someone may 'know' (e.g. *believe*) the secret 'there are weapons of mass destruction in Iraq'.

The careful reader has noted that we use the verb 'to know' in a loose way. In epistemic logic, knowledge is at least as strong as *true justified belief*.[12] When we use the verb 'to know', we technically mean 'possessing information x, which may be false'. We use 'to know' in this way because *knowledge* is an intuitive notion for the examples.

What constitutes 'comparing without disclosing' is more complicated. We will focus on *comparing information without leaking it* (CIWLI) *without reference*. What that is, and how it differs from CIWLI *with reference*, will be explaind in the remainder of this section. In zero-knowledge protocols, two players play a game in which the prover (player one) proves to the verifier (player two) that the prover has some *special knowledge*. This special knowledge could be for example knowing a Hamiltonian tour for a graph, or a password to Ali Baba's cave. The verifier (player two) does not possess the special knowledge, nor does he learn it by means of the protocol. Thus, zero-knowledge protocols are convincing but yield nothing beyond the validity of the assertion proven (in the example 'the prover knows a Hamiltonian tour') [GMR85, Gol02, BG93, BFM88].[13]

The type of knowledge that can be proven in zero-knowledge protocols is limited to knowledge within a mathematical context: the two players in a protocol know some x *a priori*, and the prover proves his knowledge of some special object y. The object x may be a public key and y the corresponding private key, or x may be a graph and y the Hamiltonian tour of it, as in the example. The required mathematical relation between x and y is, speaking loosely, that it is NP-hard to compute y from x. It might seem that the requirement of a

terrorist list is not kept secret either, as the airline carrier learns that *at least one* of the passengers on board is on the list upon being refused access to US airspace. In practice, the DHS currently discloses the identity of the suspected terrorists voluntarily to the airline carrier.

[12] Beliefs in general may be ungrounded and false. Even the definition 'true justified beliefs' has some problems [Get63].

[13] For an introduction to Zero-Knowledge protocols, consult Section 2.8.

specific mathematical relation between x and y somehow restricts the possible applications of zero-knowledge protocols.

However, it is also possible to create an NP-hard 'puzzle' on the fly to prove knowledge of any y, *provided that the verifier also knows y a priori*. If the verifier does not know y a priori, he does not gain any information which helps him to compute y. In this thesis we present the first efficient zero-knowledge protocols in which possession of *any* kind of knowledge can be proven. The knowledge need not be special in any mathematical or contextual way.[14] The assertion 'the prover knows y' can only be verified if the verifier also knows (all of) y. The verifier never learns anything more than the prover's knowledge of y, and not y itself.

This type of protocols has applications where securely comparing secrets allows transactions which could not be allowed otherwise. Examples are the comparison of police information (Section 8.1.1) and the exchange of the PNR (Section 8.1.2)

For example, secret agents might like to test each other's knowledge without exposing their own. Many examples can be found where privacy requirements or non-disclosure requirements are an obstruction for performing righteous tasks.

The type of problem that our protocols solve is similar to, but different from, the problem described in [FNW96]. We will first give a description which is broad enough to cover both problems, after which we will describe the difference.

By a secret, we mean information possessed by an agent, which the agent is not willing to share with another agent. Whether other agents indeed possess this information as well is not relevant for it being considered a secret. Here follows the problem "Comparing Information Without Leaking It" (CIWLI)[15]:

> Two players want to test whether their respective secrets are the same, but they do not want the other player to learn the secret in case the secrets do not match.

Not specified yet is *which* particular secrets are to be compared, and how it is *decided* which particular secrets are to be compared. Do the two players each take a specific secret into their mind which they compare? For example, is 'the person I voted for' equal to 'the person you voted for'? Or does one player take a secret 'The General will attack tomorrow at noon' and does the other player see whether he knows this specific secret as well? In the former case, the two players first have to agree upon what they want to compare. I call this CIWLI *with reference*. In the latter case, no a priori agreement is needed and I call it CIWLI *without reference*, because of its lack of an agreement which refers to a secret.

[14] The only requirement is that it can be uniquely encoded in a binary string, which can hardly be considered a limitation.

[15] This is a slight variation from [FNW96, page 78], where it reads "Ron and Moshe would like to determine whether the same person has complained to each of them, but, if there are two complainers, Ron and Moshe want to give no information to each other about their identities."

The difference between CIWLI with reference and CIWLI without reference can be illustrated with the following two secrets:

with reference *'I voted for Pim Fortuyn'*

> This could be a secret because it expresses a stance of the player, which he may want to keep secret for whatever reason. The reason could be fundamental (like 'votes should be secret') or practical (for example to prevent embarrassment, like admitting one still likes AᗺBA music).

without reference *'arkjjhhg bwr ufkng'*

> This could be a secret because it might be the access code to a Swiss bank account where someone keeps his fortune.

CIWLI with reference is symmetric in the sense that both players have a specific secret in mind while performing the protocol, whereas in CIWLI without reference, only one of the players has one specific secret in mind.[16]

An example of CIWLI with reference is the Socialist Millionaires' problem, in which two players want to test their riches for equality, but do not want to disclose their riches to the other player [JY96, BST01]. Another example is that two managers each have received a complaint about a sensitive matter, know this of one another, and would like to compare whether the complainer is the same person (without contacting the complainer) [FNW96]. Solutions exist for CIWLI with reference [FNW96, BST01, JY96]. In [FNW96] a series of interesting applications is listed where protocols solving this problem could be used.

It could also be the case that it is not agreed upon between the agents what the secret is about, i.e., that the agents have no particular stance towards the secret as in CIWLI with reference. In that case, we have CIWLI without reference. For example, Alice could have a file on her hard disk, and would like to know whether Bob possesses the same file as well. Alice can not naively show the file to Bob and ask him to search for a matching file, because this will obviously result in Bob obtaining the file (though Bob could be honorable and delete it voluntarily). In cases of CIWLI with reference, it is common that *two* specific secrets are tested for equality, whereas in cases without reference, one specific secret is tested against *numerous* secrets for equality. The file-comparison problem would be a case with reference if the two players would like to know whether two *specific* files are equal. ('Are the instructions you got from Carol the same as the instructions I got from Carol?')

Though secrets $f(X, Y)$ can be computed using some very complicated protocol, what will be the input X (resp. Y) remains under the control of Alice (resp. Bob). This has been acknowledged already in [Yao82, page 162]:

[16] In the field of dymanic epistemic logic, there are riddles about card deals, such as Van Ditmarsch's Russian cards problem [vD03]. It may need notice that CIWLI problems are very different from such card deal problems. Firstly, in CIWLI the number of 'cards' is unlimited (or at least extremely high), and it is not publicly known which 'cards' exist. Secondly, in CIWLI there is no such thing as *exclusive* possession of a 'card'.

> "Since a protocol can never prohibit Alice (or Bob) from behaving as if she had a different variable value X' (or Y')[17], the most that a protocol can achieve is to make sure that this is the only cheating that Alice (or Bob) can do."

In particular, a principal can always refuse to prove possession of some item, while he or she actually possesses the item. The best a protocol can achieve, is to prevent the opposite: it ensures that a principal cannot 'prove' possession of an item he does not have.

For CIWLI with reference, this is a larger problem than for CIWLI without reference. In CIWLI with reference, there is no guarantee that the input of a player is truthful (e.g., that the player *did* vote for Pim Fortuyn, or *does* like ABBA music). In CIWLI with reference, a commitment is required of both parties that their inputs to the protocol satisfy the reference, i.e., they are truthful. (For example, in the socialist millionaires' problem this means that the inputs correspond to the wealth of the players.) In fact, these protocols can only be used to test whether the two inputs are equal, and only assuming truthfulness one can say something about, for example, the riches of the players.

In CIWLI without reference, a successfully proven secret *is* truthful, because the 'truth' that is proven is the fact that the player can *construct* the secret (e.g., the access code to the Swiss bank account). However, a player can always fake *not* possessing a certain file, while he actually *does* possess the file. A player can however never fake possessing something which he does not possess (or only with negligible probability).

In this thesis, we focus on protocols for CIWLI without reference.

8.3 Adversary Models for CIWLI

In CIWLI with reference, it is required that player A cannot infer anything on the input of player B, in case their inputs do not match. This includes that it should not be possible for player A to test the input of player B for likely values, that is to guess and verify whether the guess is correct. This is called semantic security [Yao82, Yao86][18]. Semantic security is important in CIWLI with reference, because what is tested is not whether the other player can imagine or guess some input [WSI03], but whether he actually *states* the input. Thus, cases with reference should withstand guessing attacks (also called *dictionary attacks*, see Section 2.1).

In case of CIWLI without reference, there is no need to withstand guessing attacks of the players. Basically this is because cases without reference test whether the other player possesses a specific file, which is roughly equivalent to being able to imagine or guess it within the limits of its storage capacity and computational resources. In fact, the protocols we present in the next chapters

[17] In [Yao82], it says "i'" instead of "X' (or Y')" — WT

[18] Informally, an encryption scheme is semantically secure, if ciphertexts leak no information about the plaintext.

are based on the fact that a player can verify the other player's knowledge of a file by correctly 'guessing' it. Semantic security is still required in the sense that if a player cannot guess the *complete* input of the other player, he should not be able to infer *anything* of the input of the other player. And, of course, there must be full semantic security with respect to eavesdroppers, third persons other than the two players.

Given these considerations, what adversary models can best be applied?[19] In the *honest-but-curious adversary model*, the principals are supposed to adhere to protocol specification. Thus, it is assumed that in this model, the principals do not try to convince other players of possession of items which they in fact do not possess. They are only secure under the assumption of no cheating. To let go of this assumption, one has to adopt the *malicious adversary model*.

In the rest of this chapter and thesis, we will adopt the malicious adversary model.

8.4 Possible Set Relations

Algorithms for the distributed computation of set relations are tricky. For one thing, different algorithms may seem to compute the same thing, but in fact compute something different. Thus, before listing known algorithms, which we will do in the next section, it should be explained which interesting properties can be computed given two finite sets. In this section, we will explain what set relations we distinguish. All sets we consider are finite.

It is easy to characterize the possible relations between two subsets of a given domain Ω. As a reminder, two sets (each a subset of Ω) can either be:

disjoint, there is no single item that is in both sets,

partially intersecting, at least one item is found in both sets and at least one item is found in only one of the sets, or

equal, any item found in one set is also found in the other set.

The possible relations between to sets are depicted in Figure 8.1. Note that, if $X = Y = \emptyset$, the sets are both disjoint and equal. Otherwise, the relations are mutually exclusive. For example, if one determines that A and B are not equal, one can be sure that A and B are either disjoint or partially intersecting.[20]

When describing the relation between two sets, *partially intersecting* deserves some extra attention: depending on the application domain, it may or may not be required to spell out what elements constitute the partial intersection. To determine what elements constitute the intersection requires more computation than to determine whether the intersection is empty or not. To

[19] For an introduction to adversary models, see Section 2.6.

[20] For a discussion of possible set relations when one also takes the domain of possible items (e.g. the *universe*) into account, consult [KM06].

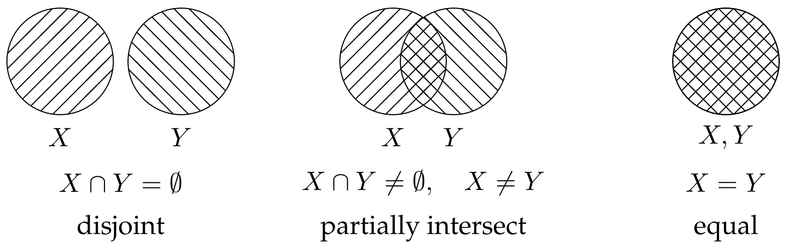

FIGURE 8.1: The relations possible between two sets X and Y.

---

disjointness Are X and Y disjoint? The answer is either a yes (1) or a no (0).

$$f_{\text{disj}}\colon \overset{<\infty}{\mathcal{P}}(\{0,1\}^*) \times \overset{<\infty}{\mathcal{P}}(\{0,1\}^*) \to \{0,1\} \quad \text{with} \quad f_{\text{disj}}(X,Y) = [X \cap Y = \emptyset]$$

intersection Which items are in both X and Y? The answer is a set.

$$f_{\text{int}}\colon \overset{<\infty}{\mathcal{P}}(\{0,1\}^*) \times \overset{<\infty}{\mathcal{P}}(\{0,1\}^*) \to \overset{<\infty}{\mathcal{P}}(\{0,1\}^*) \quad \text{with} \quad f_{\text{int}}(X,Y) = X \cap Y$$

intersection cardinality How many items are in both X and Y? The answer is a number.

$$f_{\text{ic}}\colon \overset{<\infty}{\mathcal{P}}(\{0,1\}^*) \times \overset{<\infty}{\mathcal{P}}(\{0,1\}^*) \to \mathbb{N}_0 \quad \text{with} \quad f_{\text{ic}}(X,Y) = |X \cap Y|$$

equality Are X and Y identical? The answer is either a yes (1) or a no (0).

$$f_{\text{eq}}\colon \overset{<\infty}{\mathcal{P}}(\{0,1\}^*) \times \overset{<\infty}{\mathcal{P}}(\{0,1\}^*) \to \{0,1\} \quad \text{with} \quad f_{\text{eq}}(X,Y) = [X = Y]$$

subset inclusion Is X a subset of Y? The answer is either a yes (1) or a no (0).

$$f_{\text{si}}\colon \overset{<\infty}{\mathcal{P}}(\{0,1\}^*) \times \overset{<\infty}{\mathcal{P}}(\{0,1\}^*) \to \{0,1\} \quad \text{with} \quad f_{\text{si}}(X,Y) = [X \subseteq Y]$$

This is the only relation that does not commute: $f_{\text{si}}(X,Y) \not\Leftrightarrow f_{\text{si}}(Y,X)$.

FIGURE 8.2: Some interesting set functions for which secure protocols exist.

- Items of sets (the 'secrets') are represented as bit strings.
- The set of all possible finite bit strings is represented as $\{0,1\}^*$.
- The *finite power set* (the set of all finite possible subsets) is denoted as $\overset{<\infty}{\mathcal{P}}$.
- The set of non-negative integers is denoted as $\mathbb{N}_0$.
- The cardinality of a set X is denoted as $|X|$.
- The extension of a set X is denoted as $[X]$.
- If X is a condition, then $[X] = 1$ if X holds, and $[X] = 0$ otherwise.

0-to-any At least one set is empty, say $X = \emptyset$. This case is not really interesting. In this case, for any Y, we have

$$f_{\text{disj}}(X,Y) = 1, \quad f_{\text{int}}(X,Y) = \emptyset, \quad f_{\text{ic}}(X,Y) = 0, \quad f_{\text{si}}(X,Y) = 1$$
$$\text{and} \quad f_{\text{eq}}(X,Y) = [Y = \emptyset]$$

1-to-1 Both sets contain only one element. In this case the sets are either disjoint, or equal. In this case, we have

$$f_{\text{disj}}(X,Y) = 0 \quad \Leftrightarrow \quad f_{\text{int}}(X,Y) = X \quad \Leftrightarrow \quad f_{\text{int}}(X,Y) = Y$$
$$\Leftrightarrow \quad f_{\text{ic}}(X,Y) = 1 \quad \Leftrightarrow \quad f_{\text{eq}}(X,Y) = 1 \quad \Leftrightarrow \quad f_{\text{si}}(X,Y) = 1$$

1-to-many One of the sets contains only one element, say $X = \{M\}$, and the other set Y contains more than one element. In this case Y can be disjoint with X, or partially intersecting. In this case, we have

$$f_{\text{disj}}(X,Y) = 0 \quad \Leftrightarrow \quad f_{\text{int}}(X,Y) = X$$
$$\Leftrightarrow \quad f_{\text{ic}}(X,Y) = 1 \quad \Leftrightarrow \quad f_{\text{si}}(X,Y) = 1$$

many-to-many Both sets contain more than one element. This case is not really special in the sense that there are not too many correspondences between the set relations. In this case, we only have

$$f_{\text{disj}}(X,Y) = 0 \quad \Leftrightarrow \quad f_{\text{int}}(X,Y) \neq \emptyset \quad \Leftrightarrow \quad f_{\text{ic}}(X,Y) > 0$$

FIGURE 8.3: Special cases of the sizes of two sets.

encode what elements constitute the intersection also requires more bits than to encode whether the intersection is empty or not.

Now, if one is given two finite sets X and Y, what kind of questions could one be interested to ask? In Figure 8.2, we list the most useful questions for which algorithms have been designed, with their formal definition. Without loss of generality, we assume that an item is modeled as a bit string in $\{0,1\}^*$, which contains all finite strings[21], including the empty string.

In this thesis, we will focus mainly on the *intersection* question. For more details on *intersection cardinality*, consult [FNP04, KS04]. For more details on *subset inclusion*, consult [LLM05, KM06]. In [KS04], protocols are described which determine set relation properties which are only relevant if more than two sets are involved. For the ease of comparison and discussion, we will only address the case where the number of sets to be compared is two.[22]

[21] Thus, the interpretation of $\{0,1\}^*$ is that the pattern $\{0,1\}$ may be repeated any finite number of times, but not infinitely many times.

[22] For example, in [KS04] *threshold intersection cardinality* is defined as the function that determines the set of items that each occur in more than n of the m sets, with $n > 2$ and $m > 2$.

Depending on the sizes of the sets X and Y, four cases can be distinguished. For some of these cases, the definitions of some of the questions above can be simplified, and can become the dual of one of the other definitions. Also, algorithms could be tailor-made for one of these special cases. The first case, in which one set is empty, is only included for completeness. All four cases are shown in Figure 8.3.

Clearly, the many-to-many case is the most general one. The most specific but still interesting case is 1-to-1. For the 1-to-1 case, the questions that can be asked are interchangeable.

An algorithm for the many-to-many case can be constructed by multiple runs of a 1-to-many algorithm, and an algorithm for the 1-to-many case can be constructed by multiple runs of a 1-to-1 algorithm. Usually, these are not the most efficient solutions. In Chapter 10 we show how we efficiently transform a protocol for the 1-to-many case into a protocol for the many-to-many case.

The many-to-many case is also called the *list intersection problem* [NP99]. A solution for the many-to-many case will make it possible to create indexes on distributed, secured databases, which can be searched without leaking information on the contents of the databases.[23]

8.5 Secure Protocols for Computing Set Relations

It is not very difficult to create an algorithm that computes the answer to any single question of Figure 8.2. Also, if X is only known to one principal (Alice), and Y is only known to the other principal (Bob), protocols answering these questions are easy to construct. But it is diffucult to construct protocols which compute the answers to these questions in a secure manner. In this section, we will give an overview of existing protocols for secure computation of the set relations. First, we will explain and discuss the properties that these protocols satisfy. Then, Table 8.1 lists all known, published protocols.

The protocols listed in Table 8.1 satisfy the following properties:[24]

privacy the inputs X and Y are not mutually disclosed;

validity it is impossible for the principals to cheat without the other player detecting this;

autarky the principals do not need assistance of a third party; and

efficiency for some set sizes $|X|$ and $|Y|$ the solution is efficient.[25]

[23] This is similar to, but with wider application areas than the approaches in [FLW91, FGR92].

[24] The properties privacy, validity and autarky listed here are described in more detail in Section 2.7.

[25] It would be better if all protocols are efficient for all set sizes $|X|$ and $|Y|$, but this is not the case. For some set sizes, some protocols are not even feasible. For an introduction to complexity, consult Section 2.3.

The properties *privacy* and *validity* are not easily combined with the properties *autarky* and *efficiency*. Observe that if autarky and efficiency would be dropped as requirements, there would be a trivial protocol involving a *trusted third party* (a TTP): both Alice and Bob privately disclose their inputs X and Y to the TTP, the TTP computes $f(X,Y)$, and informs Alice and Bob.

The *privacy* property is tricky in particular with respect to the *set sizes* of the inputs (i.e., $|X|$ and $|Y|$). It is easy to see that if both X and Y contain many elements, the amount of computation and comparisons required is much larger than if X and Y contain only a few elements. If it is required that $|X|$ and $|Y|$ are not disclosed, it cannot be allowed to design the protocol in such a way that it will take advantage of $|X|$ and $|Y|$ to optimize the communication and computational complexity: a protocol run would leak $|X|$ and $|Y|$ or at least some stochastic information on $|X|$ and $|Y|$. A protocol that does not disclose $|X|$ and $|Y|$ is essentially a protocol that adds dummy elements to X and Y (obtaining X' and Y') such that $|X'| = c$ and $|Y'| = c$ for some commonly agreed upper bound c. Such a protocol only works for sets with at most c elements, and 'discloses' that $|X| \leq c$ and that $|Y| \leq c$. Using such a protocol, with $c = 4 \cdot 10^6$, comparing two sets each of size 4 is just as expensive as two sets each of size 4 million. Thus, nondisclosure of $|X|$ and $|Y|$ implies a tremendous efficiency penalty.[26]

Protocols satisfying the properties listed in the beginning of this section are closely related to *secure multiparty computation* (SMC). In addition, a protocol for SMC also has to satisfy the *fairness* property. That is, both principals learn the outcome of $f(X,Y)$, and they cannot deny the other the outcome of $f(X,Y)$ without denying it oneself. The BST protocol (listed in Table 8.1) can be transformed into a fair protocol [BST01].

There is also a close relation to *zero-knowledge proofs*. If either the set sizes are a priori known to both participants, or the protocols do not leak the set sizes, the protocols can be considered zero-knowledge proofs.

Normally, in zero-knowledge proofs, it is tacitly assumed that the prover is convinced of the truth of the assertion he tries to prove. There is conceptually nothing wrong with trying to prove an assertion of which one does not know the truth value. Proving that 'I know what you know' is a very good example of this: the prover may not know the knowledge of the opponent, but can nevertheless engage in a protocol. Whether the run of this protocol yields a convincing proof depends on the knowledge of the verifier. Moreover, it is not necessary that the knowledge of the verifier is known to the prover, of course. Also, after running the protocol, the prover does not know whether the assertion proven was in fact true, or whether the proof was convincing. In [JY96], Jakobsson and Yung have introduced the concept of an *oblivious prover* which applies well to this situation. An *oblivious prover* is a prover who does not know the truth value of the assertion he tries to prove.

Now that the most important properties have been described, we are ready

[26] This has also been observed in [KM05]. Moreover, [BCC88, page 157] contains a hint to this matter: "However, Vic is not given a clue [. . .] (except perhaps for an upper limit on its length)."

name	reference	case	computes	adversary model	communication complexity	domain compression
FNW[27]	[FNW96]	1-to-1	equality[28]	various	various	no
JY	[JY96]	1-to-1	equality[28]	malicious	$\ln \lvert\Omega\rvert$	no
BST	[BST01]	1-to-1	equality[28]	malicious	$\ln \lvert\Omega\rvert$	no
NP-1	[NP99]	1-to-many	intersection[29]	malicious	$\lvert Y\rvert \cdot \ln \lvert\Omega\rvert$	no
T-1	Chapter 9	1-to-many	intersection[29]	malicious	1	yes
KM-1[30]	[KM05]	any	disjointness	honest-but-curious[31]	$\lvert\Omega\rvert \cdot \ln \lvert\Omega\rvert$	no
KM-2[30]	[KM05][32]	any	disjointness	honest-but-curious[31]	$\lvert X\rvert \cdot \lvert Y\rvert \cdot \ln \lvert\Omega\rvert$	no
KM-3[30]	[KM05][32]	any	disjointness	honest-but-curious[31]	$\binom{\lvert X\rvert+\lvert Y\rvert}{\lvert Y\rvert} \cdot \ln \lvert\Omega\rvert$	no
KM-4	[KM06]	any	disjointness	honest-but-curious[31]	$(\lvert\Omega\rvert - \lvert X\rvert) \cdot \ln \lvert\Omega\rvert$	no
NP-2	[NP99]	any	intersection	malicious	$(\lvert X\rvert + \lvert Y\rvert) \cdot \ln \lvert\Omega\rvert$	no
AgES-1[33]	[AES03]	any	intersection	honest-but-curious	$\lvert X\rvert + \lvert Y\rvert$	yes
FNP-1	[FNP04]	any	intersection	honest-but-curious	$\lvert X\rvert \cdot \ln \lvert\Omega\rvert$	no
FNP-2	[FNP04]	any	intersection	malicious	unclear	no
KS-1	[KS04]	any	intersection	honest-but-curious	$\lvert X\rvert \cdot \ln \lvert\Omega\rvert$	no
KS-2	[KS04]	any	intersection	malicious	$\lvert X\rvert^2 \cdot \ln \lvert\Omega\rvert$	no
T-2	Chapter 10	any	intersection	malicious	$\lvert X\rvert + \lvert Y\rvert$	yes
AgES-2[33]	[AES03]	any	intersection cardinality	honest-but-curious	$\lvert X\rvert + \lvert Y\rvert$	yes
FNP-3	[FNP04]	any	intersection cardinality	honest-but-curious	$\lvert X\rvert \cdot \ln \lvert\Omega\rvert$	no
KS-3	[KS04]	any	intersection cardinality	malicious	$\lvert X\rvert \cdot \ln \lvert\Omega\rvert$	no
KM-5	[KM06]	any	subset inclusion	honest-but-curious[31]	$\lvert X\rvert \cdot \lvert Y\rvert \cdot \ln \lvert\Omega\rvert$	no
KM-6	[KM06]	any	subset inclusion	honest-but-curious[31]	$(\lvert\Omega\rvert - \lvert X\rvert) \cdot \ln \lvert\Omega\rvert$	no
LLM	[LLM05]	any	subset inclusion	malicious	$\lvert\Omega\rvert \cdot \ln \lvert\Omega\rvert$	no

TABLE 8.1: All known well-documented secure protocols for computing the set relations given in Figure 8.2 (page 110).

to present the full list of protocols which securely compute set relations. It is given in Table 8.1. The following remarks may help to read the table:

- The protocols have been named after the authors of the papers in which they have been presented. Where necessary, protocols have been numbered to distinguish different protocols from the same set of authors.
- The column *case* gives the set sizes for which the protocol has been designed (see Figure 8.3). 'Any' implies any combination of set sizes, and therefore includes the many-to-many case.
- Because the properties *privacy* and *validity* can only be assessed with respect to some chosen adversary model, the adversary model is explicitly stated for each protocol.
- In the columns describing the complexity, X and Y denote the sets of Alice and Bob. The domain of all possible set items (the 'universe') is denoted Ω. The cardinality of a set Z is denoted as $|Z|$. The logarithm base 2 is denoted $\ln$. For many protocols, the communication complexity also depends on a constant factor k, a security parameter k. It has been omitted for ease of reading. Other constant factors have also been omitted.
- The column *domain compression* states whether in the protocol the items of the sets are abbreviated or not. This will be addressed further in Section 8.6.

As can be seen in Table 8.1, a lot of research has recently gone into the subject matter of secure protocols for computing set relations. Omitted from the table are protocols for which it is unclear what their security properties are [DQB95, QBA$^+$98a, QBA$^+$98b, QAD00, Ber04, CC04][34]. Also omitted from the list are 'protocols' which for their security actually rely on another protocol for computing set relations, such as [RCF04]. Another interesting algorithm is the set comparison algorithm of Wegman and Carter [WC81], which can easily

[27] FNW is only listed for completeness. In [FNW96], a total of thirteen protocols is described, some of which are only secure in the honest adversary model. Some of the protocols require physical presence or special purpose devices. None of the protocols simultaneously satisfies the privacy, validity and autarky properties. It is a valuable and enjoyable read nevertheless.

[28] In the 1-to-1 case, all interesting set relations are equivalent (see Figure 8.3). In our opinion, *equality* is the term which fits best here.

[29] In the 1-to-many case, all interesting set relations but equality are equivalent (see Figure 8.3). In our opinion, *intersection* is the term which fits best here.

[30] The protocols KM-1 through KM-3 are called PIPE #1 through PIPE #3 in [KM05].

[32] The protocols KM-2 and KM-3 are also documented in [KM06].

[31] In [KM05, KM06], it is claimed that protocols KM-1 through KM-6 can be transformed into protocols which are secure in the malicious adversary model. The proofs of these claims have not been published, nor were they available upon request.

[33] To avoid confusion with the encryption standard AES, the protocols from [AES03] have been named AgES-n. The AgES-n protocols have been somewhat modified and extended in [OYGB04], where essentially a trusted third party (TTP) is introduced.

[34] For a discussion of these articles, see Appendix C.1

name	reference	case	communication complexity	domain compression
JY	[JY96]	1-to-1	$\ln \lvert\Omega\rvert$	no
BST	[BST01]	1-to-1	$\ln \lvert\Omega\rvert$	no
NP-1	[NP99]	1-to-many	$\lvert Y\rvert \cdot \ln \lvert\Omega\rvert$	no
KS-3	[KS04]	1-to-many	$\ln \lvert\Omega\rvert$	no
T-1	Chapter 9	1-to-many	1	yes
NP-2	[NP99]	any	$(\lvert X\rvert + \lvert Y\rvert) \cdot \ln \lvert\Omega\rvert$	no
FNP-2	[FNP04]	any	unclear	no
KS-2	[KS04]	any	$\lvert X\rvert^2 \cdot \ln \lvert\Omega\rvert$	no
T-2	Chapter 10	any	$\lvert X\rvert + \lvert Y\rvert$	yes
LLM	[LLM05]	any	$\lvert\Omega\rvert \cdot \ln \lvert\Omega\rvert$	no

TABLE 8.2: Protocols which can be used for knowledge authentication This Table is a strict subset of Table 8.1, except for the KS-3 protocol, which has been restricted to the 1-to-many case (in which case *intersection cardinality* is equivalent to *subset inclusion*, see Figure 8.3).

be transformed into a protocol in the many-to-many case for computing set equality with communication complexity 1 (!). This algorithm does not belong in the table because it assumes honest principals.

All protocols which are suitable for the 'any' case leak some information on the sizes of the sets. For the 1-to-many case, the NP-1 protocol leaks the size of the bigger set, while our T-1 protocol does not. For the 1-to-1 case, the mere running of the protocol leaks that the set sizes are both equal to one, but this is rather dull information and can hardly be considered 'leaking information'.

A large number of the protocols in Table 8.1 is proven secure in the *honest-but-curious* adversary model (see Section 2.6). It is important to appreciate what this precisely means. In the honest-but-curious adversary model, the principals are supposed to adhere to protocol specification. Thus, it is assumed that in this model, the principals do not try to convince other players of possession of items which they in fact do not possess. Therefore, protocols which compute the intersection (subset inclusion, equality) problem should definitely not be considered as protocols which prove possession of set items. They are only secure under the assumption of no cheating (as discussed also in Section 8.3).

Protocols for computing intersection (subset inclusion, equality) which are secure in the *malicious* adversary model *can* be considered a proof of possession. These protocols and their main properties are summarized in Table 8.2.

8.6 Domain Compression

As can be seen in Tables 8.1 and 8.2, the communication complexities of the various protocols differ significantly. It is striking that almost all protocols have a factor $\ln|\Omega|$ in the complexity. The reason for this factor is rather simple: these protocols communicate the set items in encrypted form. If the set of items is restricted to things like secret keys, which are typically a few thousand bits long (say, 4096 bits), the domain size is $|\Omega| = 2^{4096}$. In that case $\ln|\Omega| = 4096$ bits, which is half a kilobyte[35]. With the current cost of computing power and communication bandwidth, this is by no means a prohibitive figure.

If on the other hand the set of items is not restricted to secret keys, but is extended to include binary files (say, up to sixteen megabyte), the domain size is $|\Omega| = 2^{2^{32}}$, and $\ln|\Omega| = 2^{32}$, which is sixteen megabyte. If 2^{32} is *only a factor* in the communication complexity, the feasibility of such a protocol for a domain of such size is questionable at least.

To communicate sixteen megabyte of information only to identify a single file may seem absurd, but it is necessary if communicating less information would lead to unacceptably many identification errors. When we have a set $\Phi \subsetneq \Omega$, a constant c, $|\Phi| < 2^c|\Omega|$, and Φ has a uniform distribution (for any subset of Ω), one can uniquely identify an element $x \in \Phi$ with an error probability ε of $\varepsilon = 2^{-c}$ by communicating only $c + \ln|\Phi|$ bits.

Thus, the domain size $|\Omega|$ does not impose a lower bound on the communication complexity, but $|\Phi|$, the size of the of the domain that is actually used. To obtain a lower communication complexity, we have to *compress* the *domain* Ω onto the smaller domain Φ, hence *domain compression*[36]. We need a function which provides the mapping $H\colon \Omega \to \Phi$, where the output has a uniform distribution.

A protocol that uses domain compression does not operate directly on two sets X and Y, but on two sets X' and Y' which are constructed by application of the mapping H to the sets. (Thus $X' = \{H(x)|x \in X\}$ and $Y' = \{H(y)|y \in Y\}$.)

The logical choice for a function that provides the compression, is a cryptographic hash function.[37] The output of a cryptographic hash function is indistinguishable from a uniform distribution. The output of a cryptographic hash function has a fixed length l, which is typically a few hundred for current hash functions[38], yielding a domain Φ with $|\Phi| = 2^l$. Such domain sizes for Φ are sufficiently large.

All in all, the domain Ω is compressed to a smaller domain Φ. When this is done using a cryptographic hash function, the hash values ($\in \Phi$) can be considered 'abbreviations' of the original set items ($\in \Omega$).

When it comes to computing set relations, where the items of the sets stem from a large or huge domain Ω, one can observe that for fixed sets of secrets X and Y, the larger a domain Ω, the sparser the sets will be. More specifically,

[35] Where we write ln, this is a shorthand for for $\log_2$, the logarithm with base 2.

[36] Though tempting, we refrain from using the term 'compression function', as this term also has a specific meaning in the Merkle-Damgård paradigm, which applies to cryptographic hash functions.

[37] For an extensive introduction to cryptographic hash functions, see Chapter 3.

[38] The hash function with the longest hash values known to date is SHA-512, which produces hash values of 512 bits long.

one can observe that

$$\lim_{|\Omega|\to\infty} \frac{|X| \cdot |Y|}{|\Omega|} = 0$$

which makes it abundantly clear that exploitation in the protocols of the sparsity of the sets will improve the communication complexity. The protocols which do exploit this sparsity have a 'yes' in the column 'domain compression' of Table 8.1 (they are: T-1, T-2, AgES-1 and AgES-2).[39]

The instrument that these protocols use in order to exploit the sparsity is a cryptographic hash function. Instead of communicating encrypted forms of the set items, the encrypted form of the hash value of the set items is communicated.

The use of domain compression has subtle but important implications for the *privacy* property of a protocol. First, let us repeat the definition of this property from Section 2.7. Suppose there are two principals, Alice who possesses X and Bob who possesses Y.

privacy The inputs X and Y are not mutually disclosed: Alice does not learn anything about Y except $f(X, Y)$, and Bob does not learn anything about X except $f(X, Y)$.

Domain compression implies that the privacy property of a protocol is *relaxed*. It remains the case that X and Y are not 'mutually disclosed', but in a weaker sense:

privacy′ The inputs X and Y are not mutually disclosed: for every item $y \in Y$ it holds that if Alice does not know y before the protocol, she does not know y after the protocol. Similarly for Bob: for every item $x \in X$ it holds that if Bob does not know x before the protocol, he does not know x after the protocol.

In the latter definition (of *privacy′*), it is not considered a violation of privacy if a principal can successfully mount a *dictionary attack* on the set of the other principal. In the former definition (of *privacy*), a successful dictionary attack is considered a problem.

Both definitions speak of 'nondisclosure', but this non-disclosure does not apply to exactly the same concepts. The difference between *privacy* and *privacy′* can also be explained using the distinction between the following two concepts:

the item itself the bit string which may be an element of X and/or Y

knowledge of the item the fact whether the item itself is part of the knowledge of Alice and/or Bob.

[39] In the context of sparse sets, it is appropriate to clarify an often misinterpreted result by Kalyanasundaram, Schnitger and Razborov [KS92, Raz92]. As it would distract too much from the 'story line' of this chapter, it is clarified in Appendix C.2.

In *privacy*, nondisclosure applies to both the item itself and the knowledge of the item. In *privacy*′, nondisclosure applies only to the item itself, and not to the knowledge thereof. The fact that in the definition of *privacy*′, nondisclosure does not apply to the knowledge of the item, does not mean that one can assume knowledge of the item is always disclosed. It means merely that in *privacy*′, knowledge of an item *may* be disclosed.

8.7 Conclusion

Gossiping without disclosing secrets has application in small social settings, but more importantly also in the comparison of police information (Section 8.1.1) and the exchange of the airline passenger data (Section 8.1.2).

Protocols for *knowledge authentication* are protocols that allow a player to prove possession of a secret to someone who also knows the secret, without disclosing the secret itself.

There are a number of variations of the problem, depending on the following properties:

- Does ‘secret’ mean that it is difficult to guess the string that represents secret (as with the string ‘arkjjhhg bwr ufkng’), or does it mean that the player has attributed some stance to a commonly known string? (as with ‘I voted for Pim Fortuyn’). We call the former *CIWLI without reference*, and the latter *CIWLI with reference* (see Section 8.2).

- How untrustful and untrustworthy are the players? (i.e., what *adversary model* is appropriate? see Section 8.3.)

- How many secrets need to be compared? Just one secret against one other secret (*1-to-1*), one secret against many other secrets (*1-to-many*), or many secrets against many secrets (*many-to-many*)? (see Section 8.4.)

- How many *possible secrets* exist? (i.e., what is the domain size $|\Omega|$?)
  How many *actual secrets* may exist? (i.e., what is the domain size $|\Phi|$?)
  (see Section 8.6)

There are many protocols which compute set relations in some secure manner (see Section 8.5, Table 8.1), but only a few of these protocols are suitable for *knowledge authentication* (see Table 8.2).

In the next two chapters, we will present our T-1 protocol for the 1-to-many case (Chapter 9) and our T-2 protocol for the many-to-many case (Chapter 10) . These are protocols secure in the malicious adversary model, for CIWLI without reference. These protocols use cryptographic hash functions in order to optimize the communication complexity.