Alpha-Beta Privacy

Mödersheim, Sebastian Alexander; Viganò, Luca

Link to article, DOI:
10.1145/3289255

Publication date:
2018

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
Mödersheim, S. A., \& Viganò, L. (2018). Alpha-Beta Privacy. DTU Compute. DTU Compute Technical Report2018, Vol.. 07, DOI: 10.1145/3289255

[^0]
# Alpha-Beta Privacy 

Sebastian Mödersheim ${ }^{1}$ and Luca Viganò ${ }^{2}$<br>${ }^{1}$ DTU Compute, DK-2800 Lyngby, Denmark samo@dtu.dk<br>${ }^{2}$ Department of Informatics, King's College London, UK, luca.vigano@kcl.ac.uk

DTU Technical Report-2018-7


#### Abstract

The formal specification of privacy goals in symbolic protocol models has proved to be not quite trivial so far. The most widely used approach in formal methods is based on the static equivalence of frames in the applied pi-calculus, basically asking whether or not the intruder is able to distinguish two given worlds. But then a subtle question emerges: how can we be sure that we have specified all pairs of worlds to properly reflect our intuitive privacy goal? To address this problem, we introduce in this paper a novel and declarative way to specify privacy goals, called $(\alpha, \beta)$-privacy. This new approach is based on specifying two formulae $\alpha$ and $\beta$ in first-order logic with Herbrand universes, where $\alpha$ reflects the intentionally released information and $\beta$ includes the actual cryptographic ("technical") messages the intruder can see. Then ( $\alpha, \beta$ )-privacy means that the intruder cannot derive any "non-technical" statement from $\beta$ that he cannot derive from $\alpha$ already. We describe by a variety of examples how this notion can be used in practice. Even though ( $\alpha, \beta$ )-privacy does not directly contain a notion of distinguishing between worlds, there is a close relationship to static equivalence of frames that we investigate formally. This allows us to justify (and criticize) the specifications that are currently used in verification tools, and obtain a decision procedure for a large fragment of $(\alpha, \beta)$-privacy.


## 1 Introduction

### 1.1 Context and motivation.

Over the last fifteen years or so, several formal notions of privacy in symbolic protocol models have been proposed, e.g., $[1,4,10,18,21,22,28,36,38]$ to name only a few. Although these notions are quite different, they are all witness to the fact that defining privacy is actually surprisingly subtle and not as easy as
it is sometimes thought to be. One of the main reasons is that classical secrecy notions apply only to data that are themselves the secrets, e.g., a private key. In contrast, the data that privacy goals are formulated about are typically not secrets in themselves, e.g., the name of the candidates and of the voters in a voting protocol are usually all publicly known. Rather, the information we would like to protect is the relation between these values, i.e., who voted for whom.

For this reason, the majority of the popular approaches to formalizing privacy are based not on the question of what the intruder can deduce from a set of known messages, but rather whether he can distinguish two different worlds. ${ }^{1}$ A crucial question is thus: what is the "right" set of distinguishability questions to define privacy? For instance, in voting protocols, it has become standard to define vote privacy by the following "vote swapping" encoding: take the protocol with fixed votes and consider a variant where the votes of two honest voters have been swapped; then the two variants should be indistinguishable for the intruder. We propose that this vote swap, though formally precise, is still an encoding of an unspoken underlying idea or intuition. This becomes apparent when we ask the question: is vote swapping in some sense a "correct" (sound and complete) encoding? That is, does it really capture all privacy violations that we intuitively would see as a violation and only those? In some sense we cannot completely avoid the fact that there may be a gap between intuitive ideas and mathematical notions, but simple, declarative, logical notions can give a higher confidence. Below we argue that with $(\alpha, \beta)$-privacy there is a rather straightforward formal goal we can state: that in a voting protocol the intruder only learns the number of cast votes and the result, i.e., the information that is officially published anyway. We can then prove that this is in fact equivalent to the vote-swapping trick and thereby justify this notion as being sound and complete with respect to a different, more high-level notion. In fact, this is the main theme of this paper: we do not want to replace the existing works on privacy, but we want to highlight it from a new angle that can give us novel insights and methods to reason about privacy.

It is even more difficult to come up with privacy definitions for other, more open-ended fields like identity management. In general, we have no good criterion when a given set of distinguishabilities is "complete", i.e., to be sure that we have not overlooked some possible connection the intruder could make that would violate our intuitive understanding of privacy. We return to this in the next subsection, after having introduced ( $\alpha, \beta$ )-privacy.

[^1]
### 1.2 Contributions.

In this paper, we take a step back and approach the problem of defining privacy from a different angle. Our overall contribution is the definition of a formal description that reflects the idea of privacy in a "natural" and less "technical" way. ${ }^{2}$ This allows us to formally relate such declarative logical privacy definitions with existing low-level encodings based on indistinguishability: either proving that the encoding is correct or giving a counter example (e.g., where the declarative notion of privacy is violated while the encoding does not detect it). This also allows us to use existing verification technologies, but opens a new way of understanding and declaratively using the goals, and ultimately also new ways of implementing verification tools.

The logical notion we introduce in this paper is called $(\alpha, \beta)$-privacy and it is based on specifying two formulae $\alpha$ and $\beta$ in First-Order Logic with Herbrand Universes [27].

The formula $\alpha$ formalizes the intentionally released information, i.e., the information that we consciously give to the intruder; we also refer to this information as payload. This is in fact inspired by cryptographic zero-knowledge proofs: here a prover proves a statement to the verifier (for instance, that one possesses a credential and is over 21 years old according to this credential). The verifier clearly learns this statement, including any logical consequence (for instance, that the prover is also over 18 years old). The point about zero knowledge is that the verifier does not learn any more information than the proved statement (for instance, whether or not the prover is over 65 years old). Our idea is to use such a simple formula $\alpha$ that describes the deliberately released information as the core of the privacy definition: the intruder should not be able to learn anything that is not a logical consequence of $\alpha$.

As a counterpart to the "ideal knowledge" provided by the payload $\alpha$, we also describe the technical information $\beta$, which represents the "actual knowledge" that the intruder has, describing the information (e.g., names, keys ...) that he initially knows, which actual cryptographic messages he has observed and what he knows about these messages. For instance, he may be unable to decrypt a message but anyway know that it has a certain format and contains certain (protected) information, like a vote.
$(\alpha, \beta)$-privacy then means that every "non-technical" statement that the intruder can derive from $\beta$ can already be derived from $\alpha$. We believe that this is indeed a simple way to define privacy, and is a more declarative way to talk about privacy than distinguishability of frames. To some extent, this liberates the modelers from thinking about what technical information the intruder could exploit, and rather think about only two crucial aspects of the system that they should be clear about anyway: namely, what information is deliberately released $(\alpha)$ and what messages are actually exchanged $(\beta)$.

[^2]For the example of vote privacy, the deliberately released information may be for instance the number of cast votes and for each candidate the number of received votes. We can then formally prove that this is indeed equivalent to the aforementioned vote-swapping trick. In other words, one can convince oneself by examples that the vote-swapping encoding makes sense: for instance, the intruder should not find out whether two honest voters have voted the same or differently, and if this were violated in a protocol, then this would be detected by the vote-swapping goal. However checking that a number of intuitive examples are covered is not completely satisfactory from a scientific point of view. To put it differently: the vote swap is an encoding of an (otherwise unformalized) intuition. With $\alpha-\beta$ privacy, in contrast, we are able to formally prove that it is a correct encoding of a simple privacy goal $\alpha$ : that the intruder does not find out more than we deliberately release, i.e., the number of cast votes, the election result and what the dishonest voters voted for (if that is considered).

An interesting property of $(\alpha, \beta)$-privacy is that it is stable under background information, in the following sense: if $(\alpha, \beta)$-privacy holds for a particular $\alpha$ and $\beta$, and the intruder somehow has some additional background knowledge $\alpha_{0}$, then also ( $\alpha \wedge \alpha_{0}, \beta \wedge \alpha_{0}$ )-privacy holds, i.e., the intruder cannot find out more than what follows from his background knowledge and the information that was released to him.

Another interesting and declarative feature of our approach is how we can reason about the collusion of dishonest parties. If the knowledge of every dishonest party is described by an $(\alpha, \beta)$ pair, say $\left(\alpha_{1}, \beta_{1}\right)$ and $\left(\alpha_{2}, \beta_{2}\right)$, then we can consider the conjunction of the $\alpha$ 's and the conjunction of the $\beta$ 's and require ( $\alpha_{1} \wedge \alpha_{2}, \beta_{1} \wedge \beta_{2}$ )-privacy to hold. This is indeed the strongest possible privacy goal any system can fulfill in the face of such a collusion, because we cannot prevent the dishonest parties from pooling their knowledge - both on the $\alpha$ and on the $\beta$ level-and drawing conclusions from that.

Last but not least, $(\alpha, \beta)$-privacy also provides a model-theoretic approach to privacy: given some information or observations about the world (a formula), what are the possible worlds that are compatible with this information (the models of the formula)? A privacy goal then specifies simply the set of possible worlds (the models of $\alpha$ ), and the intruder should not be able to exclude any of them using $\beta$. In fact, we will show that combining the model-theoretic and proof-theoretic views yields a powerful tool-set for reasoning about privacy.

This paper is focused on developing new foundations for reasoning about privacy and we do not discuss automated procedures. However, we prove that a fragment of $(\alpha, \beta)$-privacy corresponds to frame equivalence problems, so that existing decision procedures are applicable. $(\alpha, \beta)$-privacy is, however, more powerful and can incorporate how information is released during a protocol run. We will finally illustrate with an example how to define transition systems based on ( $\alpha, \beta$ )-privacy, i.e., where every reachable state characterizes the intruder knowledge by an $(\alpha, \beta)$ pair.

Table 1: Roadmap of the main notions and primitives introduced

| Primitive | Section | Meaning |
| :--- | :--- | :--- |
| $\Sigma, \mathcal{V}, \mathcal{T}_{\Sigma}(\mathcal{V})$ | Section 2.1, <br> p.6 (Defini- <br> tion 1) | Finite alphabet, disjoint set of variables, and terms <br> of our Herbrand Logic (FOL with Herbrand Uni- <br> verses) |
| $\Sigma_{o p}$ | Table 2, p.10 | Example set of standard cryptographic construc- <br> tors, destructors, verifiers |
| $\digamma_{i}$ | Section 2.3, <br> p.9 (Defini- <br> tion 4) | Frame (as in static equivalence), adapted to Her- <br> brand Logic |
| $\mathrm{m}_{i}$ | Section 2.3, <br> p.9 | Memory location i, storing a piece of intruder <br> knowledge |
| $\alpha$ | Section <br> p.13 | Payload, information that the intruder may legiti- <br> mately obtain, over $\mathcal{V}_{0} \subseteq \mathcal{V}$ and $\Sigma_{0} \subseteq \Sigma$ |
| $\beta$ | Section $\quad 3$, <br> p.13 | Technical information of and about observed pro- <br> tocol messages, over $\mathcal{V}$ and $\Sigma$ |
| $\phi_{\text {frame }}(\digamma)$, | Fig. 1, p.12 | Axioms used in $(\alpha, \beta)$-privacy <br> $\phi_{\digamma_{1} \sim \digamma_{2}}$ |
| concr | Section 3.2, <br> p.15 | Encoding of concrete intruder knowledge, ground <br> terms from $\mathcal{T}_{\Sigma}$ |
| struct | Section 3.2, <br> p.15 | Encoding of structural intruder knowledge, terms <br> from $\mathcal{T}_{\Sigma}(\mathcal{V})$ |

### 1.3 Organization.

Section 2 provides the basis for our approach: we discuss First-Order Logic with Herbrand Universes, messages and frames. In Section 3, we formalize $(\alpha, \beta)$ privacy and consider some concrete examples. Note that we introduce the main notions and primitives of our new ( $\alpha, \beta$ )-privacy approach step by step, where Table 1 gives an overview of where they are introduced. In Section 4, we discuss automation and the relation of $(\alpha, \beta)$-privacy to static equivalence. In Section 5 , we provide additional examples of how ( $\alpha, \beta$ )-privacy may be employed to model randomized and deterministic pooling of knowledge, and e-voting. In Section 6, we discuss how we can extend ( $\alpha, \beta$ )-privacy to transition systems. In Section 7, we show how the intruder can make use of some background knowledge to carry out attacks. Finally, in Section 8, we draw conclusions, discussing also related work and future work. This paper extends and supersedes the preliminary version [34].

## 2 Preliminaries

### 2.1 Herbrand Logic

To formalize our approach, we need to choose an appropriate logic. An obvious candidate is first-order logic ( $F O L$ ), but this has one difficulty when it comes to the interpretation of the constants and cryptographic operators. As is standard in security protocol verification, we would like to interpret these operators either in the free algebra or in the initial algebra induced by a set of algebraic equations; we call this the Herbrand Universe. ${ }^{3}$ In general, we cannot enforce the desired interpretation by axioms in FOL (see, e.g., Example 2). There are some workarounds for this; for instance, $[9,25,37,39]$ use first-order Horn theories that are inconsistent (in standard FOL) iff there is an attack in the least Herbrand model, but this construction is not possible for our work because we want to talk about deductions that hold in all Herbrand models of a formula (which does not necessarily have a unique least Herbrand model).

As proposed in [27], FOL with Herbrand universes, or Herbrand Logic for short, can be seen as a logic in its own right-as justified by the higher expressiveness, see, e.g., Example 2 below. We define Herbrand Logic as follows (and will then discuss differences with respect to the definition of [27] below).

Definition 1 (Syntax of Herbrand Logic). Let $\Sigma=\Sigma_{f} \uplus \Sigma_{i} \uplus \Sigma_{r}$ be an alphabet that consists of

- a set $\Sigma_{f}$ of free function symbols,
- a set $\Sigma_{i}$ of interpreted function symbols and
- a set $\Sigma_{r}$ of relation symbols,
all with their arities. We write $f^{n}$ and $r^{n}$ to denote a function symbol $f$ and a relation symbol $r$ of arity $n$, respectively.

We write $\mathrm{f}\left(t_{1}, \ldots, t_{n}\right)$ when $\mathrm{f} \in \Sigma_{f}$ and $f\left[t_{1}, \ldots, t_{n}\right]$ when $f \in \Sigma_{i}$, and we denote the set of considered cryptographic operators by the subset $\Sigma_{o p} \subseteq$ $\Sigma_{f}$. Constants are the special case of function symbols with arity 0 ; for an uninterpreted constant $c^{0} \in \Sigma_{f}$, we omit the parentheses and write simply $c$ instead of $c()$, whereas for interpreted constants $c^{0} \in \Sigma_{i}$, we do not omit the square brackets for clarity and write c [] .

Let $\mathcal{V}$ be a countable set of variable symbols, disjoint from $\Sigma$. We denote with $\mathcal{T}_{\Sigma}(\mathcal{V})$ the set of all terms that can be built from the function symbols in $\Sigma$ and the variables in $\mathcal{V}$. We simply write $\mathcal{T}_{\Sigma}$ when $\mathcal{V}=\emptyset$, and call its elements ground terms (over signature $\Sigma$ ). We define substitutions $\theta$ as is standard.

We define the set $\mathcal{L}_{\Sigma}(\mathcal{V})$ of formulae over the alphabet $\Sigma$ and the variables $\mathcal{V}$ as usual: relations and equality of terms are atomic formulae, and composed

[^3]formulae are built using conjunction $\wedge$, negation $\neg$, and existential quantification $\exists$.

The function fv returns the set of free variables of a formula as expected.

We employ the standard syntactic sugar and write, for example, $\forall x . \phi$ for $\neg \exists x . \neg \phi$. We also write $x \in\left\{t_{1}, \ldots, t_{n}\right\}$ to abbreviate $x=t_{1} \vee \ldots \vee x=t_{n}$.

Slightly abusing notation, we will also consider a substitution $\left\{x_{1} \mapsto t_{1}, \ldots, x_{n} \mapsto\right.$ $\left.t_{n}\right\}$ as a formula $x_{1}=t_{1} \wedge \ldots \wedge x_{n}=t_{n}$.

Definition 2 (Herbrand Universe and Algebra). Formulae in Herbrand logic are always interpreted with respect to a given fixed set $\Sigma_{f}$ of free symbols (since this set may contain symbols that do not occur in the formulae) and a congruence relation $\approx$ on $\mathcal{T}_{\Sigma_{f}}$. We may annotate all notions of the semantics with $\Sigma_{f}$ and $\approx$ when it is not clear from the context.

We write $\llbracket t \rrbracket_{\approx}=\left\{t^{\prime} \in \mathcal{T}_{\Sigma_{f}} \mid t \approx t^{\prime}\right\}$ to denote the equivalence class of a term $t \in \mathcal{T}_{\Sigma_{f}}$ with respect to $\approx$. Further, let $U=\left\{\llbracket t \rrbracket_{\approx} \mid t \in \mathcal{T}_{\Sigma_{f}}\right\}$ be the set of all equivalence classes. We call $U$ the Herbrand universe (since it is freely generated by the function symbols of $\Sigma_{f}$ modulo $\approx$ ). Based on $U$, we define a $\Sigma_{f}$-algebra $\mathcal{A}$ that interprets every $n$-ary function symbol $\mathrm{f} \in \Sigma_{f}$ as a function $f \mathcal{A}: U^{n} \rightarrow U$ in the following standard way. $\mathrm{f}^{\mathcal{A}}\left(\llbracket t_{1} \rrbracket_{\approx}, \ldots, \llbracket t_{n} \rrbracket_{\approx}\right)=\llbracket \mathfrak{f}\left(t_{1}, \ldots, t_{n}\right) \rrbracket \approx$, where the choice of the representatives $t_{1}, \ldots, t_{n}$ of the equivalence classes is irrelevant because $\approx$ is congruent. $\mathcal{A}$ is sometimes also called the quotient algebra (in the literature sometimes denoted with $\left.\mathcal{T}_{\Sigma_{f}} / \approx\right)$.

Example 1. As an example, suppose the congruence relation $\approx$ is given by a set of equations like $\forall x, y . x+y \approx y+x$ for some binary function symbol + in $\Sigma_{f}$. Then we have in the quotient algebra $5+3 \approx 3+5$ but still $3+5 \not \approx(1+2)+5$. Thus, the quotient algebra is the finest (or "free-est") interpretation still compatible with the given algebraic properties.

Definition 3 (Semantics of Herbrand Logic). An interpretation $\mathcal{I}$ maps every interpreted function symbol $f \in \Sigma_{i}$ of arity $n$ to a function $\mathcal{I}(f): U^{n} \rightarrow U$ on the Herbrand universe, every relation symbol $r \in \Sigma_{r}$ of arity $n$ to a relation $\mathcal{I}(r) \subseteq U^{n}$ on the Herbrand universe, and every variable $x \in \mathcal{V}$ to an element of $U$.

We extend $\mathcal{I}$ to a function on $\mathcal{T}_{\Sigma}(\mathcal{V})$ as expected: $\mathcal{I}\left(f\left(t_{1}, \ldots, t_{n}\right)\right)=f^{\mathcal{A}}\left(\mathcal{I}\left(t_{1}\right)\right.$, $\left.\ldots, \mathcal{I}\left(t_{n}\right)\right)$ for $\mathrm{f} \in \Sigma_{f}$ and $\mathcal{I}\left(\mathrm{f}\left[t_{1}, \ldots, t_{n}\right]\right)=\mathcal{I}(\mathrm{f})\left(\mathcal{I}\left(t_{1}\right), \ldots, \mathcal{I}\left(t_{n}\right)\right)$ for $\mathrm{f} \in \Sigma_{i}$.

We define that $\mathcal{I}$ is a model of formula $\phi$, in symbols $\mathcal{I} \models \phi$, as follows:

$$
\begin{array}{lll}
\mathcal{I} \models s=t & & \text { iff } \\
\mathcal{I}(s)=\mathcal{I}(t) \\
\mathcal{I} \models r\left(t_{1}, \ldots, t_{n}\right) & & \text { iff } \\
& \left(\mathcal{I}\left(t_{1}\right), \ldots, \mathcal{I}\left(t_{n}\right)\right) \in \mathcal{I}(r) \\
\mathcal{I} \models \phi \wedge \psi & & \text { iff } \\
\mathcal{I} \models \phi \text { and } \mathcal{I} \models \psi \\
\mathcal{I} \models \neg \phi & & \text { iff } \\
\mathcal{I} \models \exists \text { not } \mathcal{I} \models \phi \\
\models \exists & & \text { iff } \\
& \text { there is a } c \in U \text { such that } \mathcal{I}\{x \mapsto c\} \models \phi
\end{array}
$$

where $\mathcal{I}\{x \mapsto c\}$ denotes the interpretation that is identical to $\mathcal{I}$ except that $x$ is mapped to $c$. Entailment $\phi=\psi$ is defined as $\mathcal{I} \models \phi$ implies $\mathcal{I} \models \psi$ for all
interpretations $\mathcal{I}$. We write $\phi \equiv \psi$ when both $\phi \models \psi$ and $\psi \models \phi$. We also use $\equiv$ in the definitions of formulae. Finally, we write Sat $(\phi)$ if $\phi$ has a model.

For most applications, the interpretation of the Herbrand universe modulo a congruence $\approx$ is actually syntactic sugar. For instance, when $\approx$ is induced by a set of equations, it is not difficult to see that the relation $\approx$ can be axiomatized in Herbrand logic itself.

Example 2. Similar to [27], we can axiomatize arithmetic in Herbrand logic; simply let $\Sigma_{f}=\left\{z^{0}, s^{1}\right\}$, representing 0 and $(+1)$, let $\approx$ be syntactic equality on $\mathcal{T}_{\Sigma_{f}}$, and let $\Sigma_{i}=\left\{\right.$ add ${ }^{2}$, mult $\left.{ }^{2}\right\}$ and $\Sigma_{r}=\{<\}$ with the following formula:

$$
\begin{aligned}
\phi \equiv \forall x, y . & \operatorname{add}[\mathbf{z}, y]=y \wedge \operatorname{add}[\mathbf{s}(x), y]=a d d[x, \mathbf{s}(y)] \wedge \operatorname{mult}[\mathbf{z}, y]=\mathbf{z} \wedge \\
& \operatorname{mult}[\mathbf{s}(x), y]=\operatorname{add}[y, \operatorname{mult}[x, y]] \wedge x<\mathbf{s}(x) \wedge x<y \Longrightarrow x<\mathbf{s}(y)
\end{aligned}
$$

Then $\phi \models \psi$ iff $\psi$ is a true arithmetic statement. It is well-known that (as a consequence of Löwenheim-Skolem's theorem, see [23], for instance) an equivalent axiomatization cannot be achieved in standard FOL.

We note the following three differences with respect to the definition of Herbrand logic in [27]. First, in [27] and as is standard, the Herbrand universe is the free term algebra, forbidding one to model algebraic properties of the free operators. Our definition is a generalization to equivalence classes modulo the $\approx$ relation (and $\approx$ can simply be set to be the syntactic equality on $\mathcal{T}_{\Sigma_{f}}$ to get the free algebra). Second, the logic in [27] treats free variables as implicitly universally quantified, which is quite non-standard. ${ }^{4}$ In our definition, an interpretation of a formula includes the interpretation of the free variables as is standard. This is, of course, without loss of expressiveness since one can quantify variables when this is what one wants to express. Third, the logic in [27] does not have interpreted functions and, in fact, these are syntactic sugar: an interpreted $n$-ary function symbol $f$ can be modeled by an $n+1$-ary relation $R_{f}$ symbol with the axiom $\forall x_{1}, \ldots, x_{n} . \exists y . R_{f}\left(x_{1}, \ldots, x_{n}, y\right) \wedge \forall y^{\prime} . R_{f}\left(x_{1}, \ldots, x_{n}, y^{\prime}\right) \Longrightarrow y=y^{\prime}$.

### 2.2 Messages, Operators and Algebraic Properties

In $(\alpha, \beta)$-privacy we generally use a black-box algebraic model that consists of an arbitrary set of cryptographic operators and their algebraic properties. For concreteness, for most examples in this paper, we will use the set $\Sigma_{o p}$ given in Table 2.

The congruence relation $\approx$ that the Herbrand universe is defined by is the least relation so that the equations in Table 2 hold (for all terms $s, r, t, t_{1}, t_{2}$ in $\mathcal{T}_{\Sigma_{o p}}$ and for $\left.i \in\{1,2\}\right)$. Intuitively,

[^4]- $\operatorname{pub}(a)$ and $\operatorname{priv}(a)$ represent an asymmetric key pair for an agent $a$, where pub is a public function in $\Sigma_{o p}$ and priv is a private function in $\Sigma \backslash \Sigma_{o p} ;{ }^{5}$
- $\operatorname{crypt}(p, r, t)$ represents the asymmetric encryption of a message $t$ with a public key $p$ and randomness $r$;
- dcrypt $\left(p^{\prime}, t\right)$ represents the decryption with private key $p^{\prime}$ of a message $t$, and the first property formalizes that decryption with the correct private key yields the original message;
- $\operatorname{vcrypt}\left(p^{\prime}, t\right)$ and the second property formalize that we can check whether the message $t$ can be correctly decrypted with private key $p^{\prime}$ (we return to this below);
- $\operatorname{sign}\left(p^{\prime}, t\right)$, retrieve $(t)$ and $\operatorname{vsig}\left(p^{\prime}, t\right)$, together with their properties, similarly formalize digital signatures (where we here model signatures that contain the plaintext so that it can be retrieved);
- $\operatorname{scrypt}(k, t), \operatorname{dscrypt}(k, t)$ and $\operatorname{vscrypt}(k, t)$, together with their properties, similarly formalize symmetric cryptography;
- pair, $\operatorname{proj}_{i}$ and vpair, together with their properties, formalize that we assume to have a mechanism to concatenate plaintext so that it can later be decomposed in a unique way (sometimes called "serialization");
- h is a cryptographic hash function (where the lack of destructors reflects that it is hard to find a pre-image).

This model represents cryptographic implementations that allow for checking whether one has decrypted correctly (for instance, in symmetric cryptography, this is often realized by message authentication codes). This means that the real decryption functions stop with a failure whenever the message has not been encrypted with the corresponding key, and are thus partial functions. The way we model them, destructors are total functions and the corresponding verifiers represent the domain for which they are defined. Of course, we can model other cryptographic set-ups, however they often have quite different properties for privacy, e.g., the randomization is crucial in asymmetric cryptography (as illustrated by the examples below).

Note also that our model is untyped, e.g., one can build $\operatorname{crypt}\left(\mathrm{h}\left(t_{1}\right), r, t_{2}\right)$, although decryption will later fail for any term as decryption key.

### 2.3 Frames

Frames and the notion of their static equivalence are a standard way to formalize privacy goals in formal methods, e.g., [10, 18, 21]. In this paper, we define frames in a slightly non-standard way that is more convenient to formalize them

[^5]Table 2: Example set $\Sigma_{o p}$ : standard cryptographic constructors, destructors, verifiers

| Constructors | Destructors | Verifiers | Properties |
| :--- | :--- | :--- | :--- |
| pub |  |  |  |
| crypt | dcrypt | vcrypt | $\operatorname{dcrypt}(\operatorname{priv}(s), \operatorname{crypt}(\operatorname{pub}(s), r, t)) \approx t$ <br> $\operatorname{vcrypt}(\operatorname{priv}(s), \operatorname{crypt}(\operatorname{pub}(s), r, t)) \approx$ yes |
| sign | retrieve | vsig | $\operatorname{retrieve}(\operatorname{sign}(\operatorname{priv}(s), t)) \approx t$ <br> $\operatorname{vsig}(\operatorname{pub}(s), \operatorname{sign}(\operatorname{priv}(s), t)) \approx$ yes |
| scrypt | dscrypt | vscrypt | $\operatorname{dscrypt}(k, \operatorname{scrypt}(k, t)) \approx t$ <br> $\operatorname{vscrypt}(k, \operatorname{scrypt}(k, t)) \approx$ yes |
| pair | $\operatorname{proj}_{i}$ | vpair | $\operatorname{proj}\left(\operatorname{pair}\left(t_{1}, t_{2}\right)\right) \approx t_{i}$ <br> $\operatorname{vpair}\left(\operatorname{pair}\left(t_{1}, t_{2}\right)\right) \approx$ yes |
| h |  |  |  |

directly in Herbrand logic. In Section 2.4, we discuss the differences between the standard definition of frames and the one we consider here, and then, in Section 3, we relate frames to our concept of $(\alpha, \beta)$-privacy.

Definition 4 (Frame). $A$ frame is written as

$$
\digamma=\left\{\left|\mathrm{m}_{1} \mapsto t_{1}, \ldots, \mathrm{~m}_{l} \mapsto t_{l}\right|\right\},
$$

where the $\mathrm{m}_{i}$ are distinguished constants and the $t_{i}$ are ground terms that do not contain any $\mathrm{m}_{i}$. We call $\mathrm{m}_{1}, \ldots, \mathrm{~m}_{l}$ and $t_{1}, \ldots, t_{l}$ the domain and the image of the frame, respectively.

This frame represents that the intruder knows $l$ messages $t_{1}, \ldots, t_{l}$ that he can "refer to" as $\mathrm{m}_{1}, \ldots, \mathrm{~m}_{l}$. In standard Dolev-Yao-style intruder models, the intruder knowledge is just a set of messages $\left\{t_{1}, \ldots, t_{l}\right\}$; in contrast, frames give each message a unique label $\mathrm{m}_{i}$. This allows us to talk more precisely about what operations the intruder performs, e.g., "the intruder hashes the message at label $m_{1}$ and compares it with the message at label $m_{2}$ ". We may thus refer to the $\mathrm{m}_{i}$ as memory locations of the intruder knowledge. ${ }^{6}$

Definition 5 (Recipes and intruder-generable term). The set of recipes is the least set that contains $\mathrm{m}_{1}, \ldots, \mathrm{~m}_{l}$ and that is closed under all the cryptographic operators $\Sigma_{\text {op }}$. A frame $\digamma$ can be regarded as a substitution that replaces every $\mathrm{m}_{i}$ of its domain with the corresponding $t_{i}$. For a recipe $t$, we thus write $\digamma(t)$ for the term obtained by applying this substitution to $t$. An intruder generable term (or just generable term for short) is any term sor which there is a recipe $t$ with $s=\digamma(t)$.

[^6]To formalize frames and recipes (and later equivalence of frames) in Herbrand Logic, we introduce two new symbols for every distinct frame $\digamma$ that we want to talk about: an interpreted unary function symbol $k n_{\digamma}$ (for knowledge) and a unary predicate symbol $g e n_{\digamma}$ (for generate). We introduce two axioms $\phi_{\text {frame }}(\digamma)$ and $\phi_{\digamma_{1} \sim \digamma_{2}}$ that are shown in Fig. 1. Both these axioms are parameterized over a given set $\Sigma_{o p}$ of operators; for instance an expression like $\bigvee_{f^{n} \in \Sigma_{o p}} \phi$ stands for the corresponding disjunction of the instances of the formula $\phi$ for every operator $f$ of arity $n$ of $\Sigma_{o p}$.

Let, for instance, $\digamma=\left\{\left|\mathrm{m}_{1} \mapsto t_{1}, \mathrm{~m}_{2} \mapsto t_{2}, \mathrm{~m}_{3} \mapsto t_{3}, \mathrm{~m}_{4} \mapsto t_{4}\right|\right\}$ for some terms $t_{1}, \ldots, t_{4}$; then, for our example $\Sigma_{o p}$ of Table 2, we have

```
\(\phi_{\text {frame }}(\digamma) \equiv\left(\forall x \cdot\right.\) gen \(_{\digamma}(x) \Longleftrightarrow\)
    \(\left(x \in\left\{\mathrm{~m}_{1}, \mathrm{~m}_{2}, \mathrm{~m}_{3}, \mathrm{~m}_{4}\right\} \vee\right.\)
        \(\left(\exists x_{1} \cdot x=\operatorname{priv}\left(x_{1}\right) \wedge \operatorname{gen}_{\digamma}\left(x_{1}\right)\right) \vee\)
        \(\left(\exists x_{1}, x_{2}, x_{3} . x=\operatorname{crypt}\left(x_{1}, x_{2}, x_{3}\right)\right.\)
            \(\left.\wedge g e n_{\digamma}\left(x_{1}\right) \wedge \operatorname{gen}_{\digamma}\left(x_{2}\right) \wedge g e n_{\digamma}\left(x_{3}\right)\right) \vee\)
    )
)
\(\wedge\)
\(\left(\forall x_{1} \cdot \operatorname{gen}{ }_{\digamma}\left(x_{1}\right) \Longrightarrow k n_{\digamma}\left[\operatorname{priv}\left(x_{1}\right)\right]=\operatorname{priv}\left(k n_{\digamma}\left[x_{1}\right]\right)\right)\)
\(\wedge\)
\(\left(\forall x_{1}, x_{2}, x_{3} . \operatorname{gen}_{\digamma}\left(x_{1}\right) \wedge \operatorname{gen}_{\digamma}\left(x_{2}\right) \wedge \operatorname{gen}_{\digamma}\left(x_{3}\right) \Longrightarrow\right.\)
    \(\left.k n_{\digamma}\left[\operatorname{crypt}\left(x_{1}, x_{2}, x_{3}\right)\right]=\operatorname{crypt}\left(k n_{\digamma}\left[x_{1}\right], k n_{\digamma}\left[x_{2}\right], k n_{\digamma}\left[x_{3}\right]\right)\right)\)
    \(\wedge \ldots\)
```

The formula $\phi_{\text {frame }}(\digamma)$ characterizes a frame in Herbrand logic as follows. The first conjunct defines the predicate $g e n_{\digamma}$ to be exactly the set of recipes for the frame $\digamma=\left\{\mathrm{m}_{1} \mapsto t_{1}, \ldots, \mathrm{~m}_{l} \mapsto t_{l}\right\}$; the second and third conjuncts formalize that $k n_{\digamma}[t]$ is the result of applying recipe $t$ to frame $\digamma$, i.e., replacing every occurrence of a label $\mathrm{m}_{i}$ by the corresponding $t_{i}$ in $t$. Thus, we have:

$$
\begin{aligned}
\mathcal{I} \models & \phi_{\text {frame }}(\digamma) \text { iff } \\
& \mathcal{I}\left(\text { gen }_{\digamma}\right)=\left\{[t]_{\approx} \mid t \in \mathcal{T}_{\Sigma_{o p} \cup\left\{\mathbf{m}_{1}, \ldots, \mathrm{~m}_{l}\right\}}\right\} \text { and } \\
& \left.\mathcal{I}\left(k n_{\digamma}\right)\left([t]_{\approx)}\right)=[\digamma(t)] \approx \text { for every } t \in \mathcal{T}_{\Sigma_{o p} \cup\left\{\mathrm{~m}_{1}, \ldots, \mathrm{~m}_{l}\right\}}\right\}
\end{aligned}
$$

Example 3. Consider the frame (from [18]):

$$
\digamma_{1}=\left\{\mathrm{m}_{1} \mapsto \operatorname{scrypt}\left(\mathrm{k}, \mathrm{n}_{1}\right), \mathrm{m}_{2} \mapsto \operatorname{pair}\left(\mathrm{n}_{1}, \mathrm{n}_{2}\right), \mathrm{m}_{3} \mapsto \mathrm{k} \mid\right\} .
$$

The intruder can then, e.g., obtain $\mathrm{n}_{1}$ as follows: let $\Phi \equiv \phi_{\text {frame }}\left(\digamma_{1}\right)$; then, $\Phi \models g e n_{\digamma_{1}}\left(\operatorname{dscrypt}\left(\mathrm{~m}_{3}, \mathrm{~m}_{1}\right)\right) \wedge k n_{\digamma_{1}}\left[\operatorname{dscrypt}\left(\mathrm{~m}_{3}, \mathrm{~m}_{1}\right)\right]=\mathrm{n}_{1}$. We then also have $\Phi \models k n_{\digamma_{1}}\left[\operatorname{dscrypt}\left(\mathrm{~m}_{3}, \mathrm{~m}_{1}\right)\right]=k n_{\digamma_{1}}\left[\operatorname{proj}_{1}\left(\mathrm{~m}_{2}\right)\right]$, which intuitively means that the intruder can check that the decrypted term is equal to the first component of the term in $\mathrm{m}_{2}$.

The main idea behind static equivalence of frames (e.g., $[10,18,21]$ ) is to ask whether the intruder is able to detect the difference between two "intruder

For every considered frame $\digamma=\left\{\mid m_{1} \mapsto t_{1}, \ldots, m_{l} \mapsto t_{l}\right\}$, let $k n_{\digamma}$ be an interpreted unary function symbol and $g e n_{\digamma}$ be a unary predicate.
For a frame $\digamma$ :

$$
\begin{aligned}
\phi_{\text {frame }}(\digamma) \equiv & \left(\forall x . \operatorname{gen_{\digamma }}(x) \Longleftrightarrow\right. \\
& \left(x \in\left\{\mathrm{~m}_{1}, \ldots, \mathrm{~m}_{l}\right\} \vee\right. \\
& \left.\left.\quad \bigvee_{\mathrm{f}^{n} \in \Sigma_{o p}} \exists x_{1} \ldots x_{n} . x=\mathrm{f}\left(x_{1}, \ldots, x_{n}\right) \wedge \operatorname{gen}_{\digamma}\left(x_{1}\right) \wedge \ldots \wedge \operatorname{gen}_{\digamma}\left(x_{n}\right)\right)\right) \\
& \left(k n_{\digamma}\left[\mathrm{m}_{1}\right]=t_{1} \wedge \ldots \wedge k n_{\digamma}\left[\mathrm{m}_{l}\right]=t_{l}\right) \\
& \wedge \\
& \bigwedge_{\mathrm{f} n \in \Sigma_{o p}}\left(\forall x_{1} \ldots x_{n} .\right. \\
& \left.\operatorname{gen}_{\digamma}\left(x_{1}\right) \wedge \ldots \wedge g e n_{\digamma}\left(x_{n}\right) \Longrightarrow k n_{\digamma}\left[\mathrm{f}\left(x_{1}, \ldots, x_{n}\right)\right]=\mathrm{f}\left(k n_{\digamma}\left[x_{1}\right], \ldots, k n_{\digamma}\left[x_{n}\right]\right)\right)
\end{aligned}
$$

For two frames $\digamma_{1}$ and $\digamma_{2}$ :

$$
\begin{aligned}
\phi_{\digamma_{1} \sim \digamma_{2}} \equiv & \left(\forall x \cdot \operatorname{gen}_{\digamma_{1}}(x) \Longleftrightarrow \operatorname{gen}_{\digamma_{2}}(x)\right) \\
& \wedge \\
& \left(\forall x, y \cdot g e n_{\digamma_{1}(x)} \wedge \operatorname{gen}_{\digamma_{1}(y)} \Longrightarrow\left(k n_{\digamma_{1}}[x]=k n_{\digamma_{1}}[y] \Longleftrightarrow k n_{\digamma_{2}}[x]=k n_{\digamma_{2}}[y]\right)\right)
\end{aligned}
$$

Figure 1: Axioms used in $(\alpha, \beta)$-privacy parameterized over a given set $\Sigma_{o p}$ of operators.
knowledges" (that have the same domain): can the intruder make any check on the knowledge, i.e., two recipes so that one frame yields the same message for both recipes, while the other frame yields different messages? If there is no such critical pair of recipes, i.e., if the intruder has no way to tell whether he is "in" one frame or the other, then the frames are statically equivalent. We can formalize this in Herbrand logic using the axiom $\phi_{\digamma_{1} \sim \digamma_{2}}$ :

Definition 6 (Static Equivalence of Frames). Two frames $\digamma_{1}$ and $\digamma_{2}$ with the same set $\left\{\mathrm{m}_{1}, \ldots \mathrm{~m}_{l}\right\}$ of memory locations are statically equivalent (in symbols, $\left.\digamma_{1} \sim \digamma_{2}\right)$ iff $\operatorname{Sat}\left(\phi_{\text {frame }}\left(\digamma_{1}\right) \wedge \phi_{\text {frame }}\left(\digamma_{2}\right) \wedge \phi_{\digamma_{1} \sim \digamma_{2}}\right)$.

Example 4. We can distinguish $\digamma_{1}$ of Example 3 from the frame

$$
\digamma_{2}=\left\{\left|\mathrm{m}_{1} \mapsto \operatorname{scrypt}\left(\mathrm{k}, \mathrm{n}_{3}\right), \mathrm{m}_{2} \mapsto \operatorname{pair}\left(\mathrm{n}_{1}, \mathrm{n}_{2}\right), \mathrm{m}_{3} \mapsto \mathrm{k}\right|\right\}
$$

since the check $k n_{\digamma_{2}}\left[\operatorname{dscrypt}\left(\mathrm{~m}_{3}, \mathrm{~m}_{1}\right)\right]=k n_{\digamma_{2}}\left[\operatorname{proj}_{1}\left(\mathrm{~m}_{2}\right)\right]$ fails, whereas the same check succeeds for $k n_{\digamma_{1}}$.

### 2.4 Differences with Respect to the Standard Definition of Frames

The standard definition of frames is of the form $\nu n_{1}, \ldots, n_{k} . \theta$, where $\theta$ is a substitution from variables $x_{1}, \ldots, x_{l}$ to terms $t_{1}, \ldots, t_{l}$, respectively, and where the $x_{i}$ do not occur in the $t_{i}$. (We use the distinct constants $m_{\mathrm{i}}$ instead of the
variables $x_{i}$.) Further, the $n_{i}$ are the restricted names. Intuitively, the intruder knows all names that occur freely, i.e., that are not under a restriction. Note that in the frame $\nu n .\left\{m_{1} \mapsto n\right\}$ the intruder can produce $n$ anyway, even though it is restricted, since he has it as a message in his knowledge. In contrast, in our notion of frames, all constants are by default unknown to the intruder; thus, public constants must be modeled in our framework by putting them into the frame explicitly. This is of course not a restriction (if the set of public constants is finite), it only may mean longer frames. While the notion of restricted names is very handy in process calculi, they do not really fit well with a logical approach, since they are a mixture between constants and variables.

## 3 A New Privacy Model: $(\alpha, \beta)$-privacy

We introduce $(\alpha, \beta)$-privacy step by step. In Section 3.1, we introduce the distinction between payload formulae $\alpha$ and technical formulae $\beta$ as well as the notion of interesting consequences. In Section 3.2, we establish the methodology to reason over such formulae. We also define what it means for $\alpha$ to be combinatoric and what is a message-analysis problem. In Section 3.3, we show how $(\alpha, \beta)$-privacy extends straightforwardly to the case of dishonest agents pooling their knowledge. In Section 4, we discuss automation and the relation to static equivalence. We then discuss further examples of $(\alpha, \beta)$-privacy in Section 5. In Section 6, we discuss how to extend the $(\alpha, \beta)$-privacy notion to transition systems. In Section 7, we consider the presence of additional background knowledge.

### 3.1 Payload and Technical Information

Our model is inspired by zero-knowledge proofs for privacy (as they are used, e.g., in IBM's Idemix [28]). The following points are characteristic for such proofs:

- The prover (intentionally) conveys some information to the verifier, i.e., the statement being proved to the verifier. We call this statement the payload $\alpha$.
- The participants also (inevitably) convey some cryptographic information (e.g., commitments, challenges, and responses) that, if the scheme is secure, do not reveal anything "interesting" besides $\alpha$; this, of course, is the very reason why such a scheme is called zero-knowledge. We call this kind of information the technical information $\beta$.

The term "interesting" that we use here is often defined in the cryptography world by the fact that it is computationally easy to produce a fake transcript of zero-knowledge proofs that is statistically indistinguishable from a real transcript. Hence, whatever information could possibly be obtained from $\beta$, one may have created oneself. This kind of definition is, however, quite unhandy in logical reasoning, and it applies only to (some types of) zero-knowledge proofs.

We show that it is fortunately possible to define the term "interesting" on a logical basis that makes sense for many actual situations in which we want to talk about privacy. The key idea is that the payload $\alpha$ may be formulated over a restricted alphabet $\Sigma_{0} \subsetneq \Sigma$, whereas the technical information $\beta$ may talk about the full alphabet $\Sigma$ (e.g., all cryptographic operators are part of $\Sigma \backslash \Sigma_{0}$ ). Hence, we can define ( $\alpha, \beta$ )-privacy as follows.

Definition 7 (Interesting consequences and respect/violation of privacy). Let $\Sigma_{0} \subsetneq \Sigma$. Given a payload formula $\alpha \in \mathcal{L}_{\Sigma_{0}}(\mathcal{V})$ and a technical formula $\beta \in$ $\mathcal{L}_{\Sigma}(\mathcal{V})$, where $\beta=\alpha$ and $f v(\alpha)=f v(\beta)$ and both $\alpha$ and $\beta$ are consistent, we say that a statement $\alpha^{\prime} \in \mathcal{L}_{\Sigma_{0}}(f v(\alpha))$ is an interesting consequence of $\beta$ (with respect to $\alpha$ ) if $\beta \neq \alpha^{\prime}$ but $\alpha \not \vDash \alpha^{\prime}$.

We say that $\beta$ respects the privacy of $\alpha$ if it has no interesting consequences, and that $\beta$ violates the privacy of $\alpha$ otherwise.

Before we move on to the discussion of privacy on messages, let us say a few more words on the intuition behind $\Sigma_{0}, \alpha, \alpha^{\prime}$ and $\beta . \Sigma_{0}$ is a sub-alphabet of $\Sigma$, namely the restricted alphabet over which we can define the payloads $\alpha$ and the interesting consequences $\alpha^{\prime}$. The choice of $\Sigma_{0}$ depends, of course, on the specific case study we are considering, and we will give a number of examples in the next subsection and in Section 5. $\Sigma_{0}$ allows us to write the statements $\alpha$ that will be revealed to the intruder and their consequences $\alpha^{\prime}$. As such, $\Sigma_{0}$ will typically contain "payload" information, i.e., the information a system actually deals with, as opposed to all cryptographic functions and communications that are used to implement the system. For instance, in a voting system, $\Sigma_{0}$ could contain the names of three electoral candidates a, b and c , and $\alpha$ could then "reveal" that the actual vote being cast is $x \in\{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$, as opposed to the specific cryptographic messages being sent in the system that use hash functions and cryptographic operators to encrypt the votes. Note also that, given the importance of $\Sigma_{0}$, we could actually write " $(\alpha, \beta)$-privacy with respect to $\Sigma_{0} "$. However, we take the liberty to write just $(\alpha, \beta)$-privacy as it is simpler and $\Sigma_{0}$ is, almost always, clear from context.

We have defined the notion of an interesting consequence $\alpha^{\prime}$ as anything the intruder may be able to derive from his knowledge $\beta$ as long as it is a non-technical statement (i.e., of $\mathcal{L}_{\Sigma_{0}}$ ) and it does not follow from $\alpha$ alone, i.e., from what he is permitted to know anyway. This allows us to capture that the intruder may well see a few technical details, e.g., that two messages come from the same IP address, but that in itself is not very interesting or useful as long as he cannot tie that to a relevant information $\alpha^{\prime}$.

Another aspect of this definition is that by the information $\alpha$ that we gave out, also all information that can be derived from $\alpha$ is given out, because the best cryptographic systems cannot prevent the intruder from drawing conclusions (by making derivations from the cryptographic messages he knows). In general, the weaker $\alpha$ is (i.e., the less information we deliberately release to the intruder) and the stronger $\beta$ is (i.e., the more information we assume the intruder might actually have), the stronger is the notion of privacy. So, as a rule of thumb, when in doubt, one should be restrictive on $\alpha$ and generous on $\beta$.

In the examples below, we will see that, at least for easy systems, it is actually possible to "infer" $\beta$ systematically from the definition of the system under study: $\beta$ is namely defined as the conjunction of $\alpha$ and the instantiation of the axioms $\phi_{\text {frame }}(\digamma)$ and $\phi_{\digamma_{1} \sim \digamma_{2}}$ for the specific system, including in particular its specific set $\Sigma_{o p}$ of cryptographic operators and the cryptographic messages exchanged. One could even compute $\beta$ from a formal system specification of the system written, e.g., in the applied pi-calculus or in an extended Alice-Bob notation such as [6]. In general, however, it might not be possible to systematically infer $\beta$ from the system, as $\beta$ may actually contain information that goes beyond the exchanged messages, such as timing information, for instance.

### 3.2 Privacy on Messages

The frames that we have introduced above formalize the knowledge of the intruder. The new idea is that we also model the structural information that the intruder has about messages, and that that information takes the form of a frame too, but with variables in messages. The concrete knowledge (that is modeled by frames so far) is then an instance of this structural knowledge. Thus, a key idea of $(\alpha, \beta)$-privacy is to make explicit the fact that the intruder will usually also have information about the structure of messages and can use this for his reasoning.

For instance, the intruder may know a message $f(a, a)$, and he may know that the term is the application of the operator $f$ to two terms, but he may not know which terms. This structural knowledge can be expressed by the term $f(x, y)$ with two variables $x$ and $y$ as placeholders for what the intruder cannot determine so far. If the intruder learned that the two unknown arguments are the same, then we would have the structural information $f(x, x)$.

For most part of this paper, we will call the frame for the structural information struct and the frame for the concrete knowledge concr, and for simplicity we will abuse notation and write also struct [•] and concr $[\cdot]$ in Herbrand logic instead of $k n_{\text {struct }}[\cdot]$ and $k n_{\text {concr }}[\cdot]$. Since struct and concr will then also have the same domain, it follows that gen $_{\text {struct }}$ and $g e n_{\text {concr }}$ are equivalent and thus we will simply write $g e n$.

The variables that occur in frame struct are the free variables from $\alpha$, e.g., the intruder may know about an encrypted term that it contains a particular secret, although he does not know this secret.

The axiom $\phi_{\text {frame }}(\digamma)$ can now be instantiated with concr and struct. This formalizes that (a) the intruder knowledge is closed under application of public operators and (b) when the intruder composes terms himself, he knows the structure of the result as far as he knows the structure of the subterms.

An interesting question is now what it means if we also instantiate the axiom $\phi_{\digamma_{1} \sim \digamma_{2}}$ with concr and struct, i.e., $\phi_{\text {concr } \sim \text { struct }}$. This expresses that the intruder can connect knowledge of concrete terms and their structure: two recipes yield the same intruder-generable messages iff they have the same structure.

Let us now illustrate this by introducing a running example. In $(\alpha, \beta)$ privacy, we typically consider infinite state-transition systems and we now first
focus on the analysis of privacy with respect to one particular reachable state of such a system.

Example 5 (A simple voting example). As an example, let us consider a very simplistic toy e-voting protocol (we will discuss more realistic examples in Section 5 and Section 6). Assume that users vote by choosing values $x_{i}$ from a payload alphabet $\Sigma_{0}=\{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$, and that, as part of the protocol, a voting server publishes messages of the form $\mathrm{h}\left(x_{i}\right)$. Consider an intermediate state of an election in which only one vote $x=$ a has been cast (and the final result of the election has not yet been released by the server). Then, $\alpha$ and $\beta$ for this state could look as follows:

$$
\begin{aligned}
& \alpha \equiv x \in\{\mathrm{a}, \mathrm{~b}, \mathrm{c}\} \\
& \beta \equiv \alpha \wedge \phi_{\text {frame }}(\text { concr }) \wedge \phi_{\text {frame }}(\text { struct }) \wedge \phi_{\text {concr } \sim \text { struct }}
\end{aligned}
$$

for a frame struct $=\left\{\mathrm{m}_{1} \mapsto \mathrm{a}, \mathrm{m}_{2} \mapsto \mathrm{~b}, \mathrm{~m}_{3} \mapsto \mathrm{c}, \mathrm{m}_{4} \mapsto \mathrm{~h}(x)\right\}$, so that concr $=$ $\theta($ struct ) for a substitution $\theta=\{x \mapsto \mathrm{a}\}$. Then, in particular, we have that

$$
\begin{aligned}
& \operatorname{concr}\left[\mathrm{m}_{1}\right]=\operatorname{struct}\left[\mathrm{m}_{1}\right]=\mathrm{a} \wedge \\
& \operatorname{concr}\left[\mathrm{~m}_{2}\right]=\operatorname{struct}\left[\mathrm{m}_{2}\right]=\mathrm{b} \wedge \\
& \operatorname{concr}\left[\mathrm{~m}_{3}\right]=\operatorname{struct}\left[\mathrm{m}_{3}\right]=\mathrm{c} \wedge \\
& \operatorname{concr}\left[\mathrm{~m}_{4}\right]=\mathrm{h}(\mathrm{a}) \wedge \operatorname{struct}\left[\mathrm{m}_{4}\right]=\mathrm{h}(x)
\end{aligned}
$$

The payload formula $\alpha$ expresses the obvious, namely, that the intruder knows that the vote is in $\Sigma_{0}$. The technical formula $\beta$ contains $\alpha$, the concrete and structural knowledge, and the ability to compare them. $\beta$ thus expresses the fact that the intruder knows

- all the constants of $\Sigma_{0}$ (in memory locations $\mathrm{m}_{1}, \mathrm{~m}_{2}$ and $\mathrm{m}_{3}$, for which the structural information is identical to the concrete information),
- the concrete (observed) hash of the vote (in $\mathrm{m}_{4}$ ) and the structural information that the vote message has the form $\mathrm{h}(x)$.

We, the modelers, can indirectly read off from the formula $\beta$ "what actually happened": there is only one interpretation of the $x$ such that concr $\left[\mathrm{m}_{4}\right]=$ struct $\left[\mathrm{m}_{4}\right]$ holds, namely $x=\mathrm{a}$. The intruder cannot reason this way, and thus in general will not know the right interpretation of the variables, but may in some cases be able to infer it by comparing concrete and structural information he has, as is the case in this example: first, observe that, by $\phi_{\text {frame }}($ concr $)$, we have concr $\left[\mathrm{h}\left(\mathrm{m}_{1}\right)\right]=$ concr $\left[\mathrm{m}_{4}\right]$; hence, by applying $\phi_{\text {concr } \sim \text { struct }}$, we get struct $\left[\mathrm{h}\left(\mathrm{m}_{1}\right)\right]=\operatorname{struct}\left[\mathrm{m}_{4}\right]$, which, by $\phi_{\text {frame }}($ struct $)$, yields $\mathrm{h}(\mathrm{a})=\mathrm{h}(x)$; thus, we conclude that the intruder can indeed find out that the vote was $x=\mathrm{a}$, meaning that $\beta$ does not respect the privacy of $\alpha$ (with respect to $\Sigma_{0}=\{a, b, c\}$ ) and that the protocol is not safe.

Let us therefore consider a more refined protocol by adding a fixed and secret number n : the voting server now publishes messages of the form $\mathrm{h}\left(\operatorname{pair}\left(\mathrm{n}, x_{i}\right)\right)$ for a number n known only to the server, i.e., n is a secret from $\Sigma \backslash \Sigma_{0}$. (Obviously,
using such a fixed number, even though secret from the intruder, is a risk for guessing attacks, but we will abstract away from guessing for now and discuss it later.) Let us again consider an intermediate state of the election in which only one vote $x=$ a has been cast (and the final result of the election has not yet been released by the server). Then, the $\alpha$ and $\beta$ for this state could look as follows:

$$
\begin{aligned}
\alpha & \equiv x \in\{\mathrm{a}, \mathrm{~b}, \mathrm{c}\} \\
\beta & \equiv \alpha \wedge \phi_{\text {frame }}(\text { concr }) \wedge \phi_{\text {frame }}(\text { struct }) \wedge \phi_{\text {concr } \sim \text { struct }}
\end{aligned}
$$

for a frame struct $=\left\{\left|\mathrm{m}_{1} \mapsto \mathrm{a}, \mathrm{m}_{2} \mapsto \mathrm{~b}, \mathrm{~m}_{3} \mapsto \mathrm{c}, \mathrm{m}_{4} \mapsto \mathrm{~h}(\operatorname{pair}(\mathrm{n}, x))\right|\right\}$, with concr $=\theta$ (struct) for $\theta=\{x \mapsto \mathrm{a}\}$. Now, $\beta$ expresses that the intruder knows the observed hash of the vote (concr $\left[\mathrm{m}_{4}\right]=\mathrm{h}(\operatorname{pair}(\mathrm{n}, \mathrm{a}))$ and the structural information that this message has the form struct $\left[\mathrm{m}_{4}\right]=\mathrm{h}(\operatorname{pair}(\mathrm{n}, x))$. Here, the intruder does not know n and thus $\beta$ has several satisfying interpretations of the free variables, i.e., $\beta$ has models for each $x \in\{a, b, c\}$ ), which represents the uncertainty of the intruder. Hence, this $\beta$ does respect the privacy of $\alpha$ (with respect to $\left.\Sigma_{0}=\{\mathrm{a}, \mathrm{b}, \mathrm{c}\}\right)$.

One would thus be tempted to say that the variant of the protocol with the fixed secret number is safe, but what happens if there are some dishonest voters who collaborate with the intruder? Let us now focus on an intermediate state of the election in which only two votes $x_{1}$ and $x_{2}$ have been cast (and the final result of the election has not yet been released by the server), where $x_{1}=\mathrm{a}$ is from $a$ dishonest voter and $x_{2}=\mathrm{b}$ from an honest one. Then, the $\alpha$ and $\beta$ for this state could look as follows:

$$
\begin{aligned}
& \alpha \equiv x_{1}, x_{2} \in\{\mathrm{a}, \mathrm{~b}, \mathrm{c}\} \wedge x_{1}=\mathrm{a} \\
& \beta \equiv \alpha \wedge \phi_{\text {frame }}(\text { concr }) \wedge \phi_{\text {frame }}(\text { struct }) \wedge \phi_{\text {concr } \sim \text { struct }}
\end{aligned}
$$

for a frame struct $=\left\{\mathrm{m}_{1} \mapsto \mathrm{a}, \mathrm{m}_{2} \mapsto \mathrm{~b}, \mathrm{~m}_{3} \mapsto \mathrm{c}, \mathrm{m}_{4} \mapsto \mathrm{~h}\left(\operatorname{pair}\left(\mathrm{n}, x_{1}\right)\right), \mathrm{m}_{5} \mapsto\right.$ $\mathrm{h}\left(\operatorname{pair}\left(\mathrm{n}, x_{2}\right)\right\}$, with concr $=\theta$ (struct) for $\theta=\left\{x_{1} \mapsto \mathrm{a}, x_{2} \mapsto \mathrm{~b}\right\}$, where now $\alpha$ expresses that the intruder knows that both votes are in $\Sigma_{0}$ and that he knows the value of the dishonest vote, whereas $\beta$ expresses that the intruder not only knows the concrete observed messages $\mathrm{h}(\operatorname{pair}(\mathrm{n}, \mathrm{a}))$ and $\mathrm{h}(\operatorname{pair}(\mathrm{n}, \mathrm{b}))$, which are respectively in $\mathrm{m}_{4}$ and $\mathrm{m}_{5}$, but he also has the structural information that these messages have the form $\mathrm{h}\left(\operatorname{pair}\left(\mathrm{n}, x_{i}\right)\right)$.

The intruder can now reason as follows: since, by $\phi_{\text {frame }}($ concr $)$, it holds that concr $\left[\mathrm{m}_{4}\right] \neq$ concr $\left[\mathrm{m}_{5}\right]$ (recall that all terms are interpreted in the Herbrand universe), we have struct $\left[\mathrm{m}_{4}\right] \neq \operatorname{struct}\left[\mathrm{m}_{5}\right]$ by $\phi_{\text {concr } \sim \text { struct }}$, so that $\mathrm{h}\left(\operatorname{pair}\left(\mathrm{n}, \mathrm{x}_{1}\right)\right) \neq$ $\mathrm{h}\left(\operatorname{pair}\left(\mathrm{n}, x_{2}\right)\right)$, and therefore $x_{1} \neq x_{2}$ (again since terms are interpreted in the Herbrand universe). Since the intruder knows already the dishonest vote $x_{1}=a$, he knows $x_{2} \neq a$, and can hence derive from $\beta$ the $\Sigma_{0}$-formula $\alpha^{\prime} \equiv x_{2} \in\{\mathrm{~b}, \mathrm{c}\}$ that does not follow from $\alpha$. Thus, in this example, $\beta$ does not respect the privacy of $\alpha$ (with respect to $\Sigma_{0}=\{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$ ). Note that the intruder cannot derive more, which is-very declaratively -because $\beta$ has both a model in which $x_{2}=\mathrm{b}$, and one in which $x_{2}=\mathrm{c}$, so the intruder was not even able to determine the
choice $x_{2}$, he was only able to exclude one interpretation, namely $x_{2}=\mathrm{a} .{ }^{7}$
Let us briefly also consider three further variants of the example with $\alpha \equiv x \in$ $\{a, b, c\}$. First, if the intruder also knows n , say $\operatorname{concr}\left[\mathrm{m}_{6}\right]=\operatorname{struct}\left[\mathrm{m}_{6}\right]=\mathrm{n}$, then he can indeed derive $x=\mathrm{b}$, because he can verify that $\mathrm{h}\left(\operatorname{pair}\left(\mathrm{m}_{6}, \mathrm{~m}_{2}\right)\right)$ gives the same concrete value as $\mathrm{m}_{4}$.

Second, if the server uses different secret nonces instead of a fixed number for all votes, i.e., $\beta \equiv \ldots$ concr $\left[\mathrm{m}_{4}\right]=\operatorname{struct}\left[\mathrm{m}_{4}\right]=\mathrm{h}\left(\operatorname{pair}\left(\mathrm{n}_{1}, \mathrm{a}\right)\right) \wedge \operatorname{concr}\left[\mathrm{m}_{5}\right]=$ $\mathrm{h}\left(\operatorname{pair}\left(\mathrm{n}_{2}, \mathrm{~b}\right)\right) \wedge \operatorname{struct}\left[\mathrm{m}_{5}\right]=\mathrm{h}\left(\operatorname{pair}\left(\mathrm{n}_{2}, x\right)\right)$, then $\beta$ indeed preserves the privacy of $\alpha$. To see this, note that $\beta$ has models with $x=\mathrm{a}$, with $x=\mathrm{b}$, and with $x=\mathrm{c}$. Therefore, every $\Sigma_{0}$-formula $\alpha^{\prime}$ that follows from $\beta$ also follows from $\alpha$ and thus $\beta$ respects the privacy of $\alpha$ (with respect to $\Sigma_{0}=\{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$ ).

Third, so far $\mathrm{m}_{4}$ represented the vote of a dishonest agent and the intruder knew already its value $x_{1}=$ a. Protecting the privacy of two honest votes can be formalized as follows (where we have two different nonces, but model them just as some fixed constants that the intruder does not know by default):

$$
\begin{aligned}
& \alpha \equiv x_{1} \in\{\mathrm{a}, \mathrm{~b}, \mathrm{c}\} \wedge x_{2} \in\{\mathrm{a}, \mathrm{~b}, \mathrm{c}\} \\
& \beta \equiv \alpha \wedge \phi_{\text {frame }}(\text { concr }) \wedge \phi_{\text {frame }}(\text { struct }) \wedge \phi_{\text {concr } \sim \text { struct }}
\end{aligned}
$$

such that $\beta$ includes

$$
\begin{aligned}
\ldots \quad \text { concr }\left[\mathrm{m}_{4}\right] & =\mathrm{h}\left(\operatorname{pair}\left(\mathrm{n}_{1}, \mathrm{a}\right)\right) \wedge \operatorname{struct}\left[\mathrm{m}_{4}\right]=\mathrm{h}\left(\operatorname{pair}\left(\mathrm{n}_{1}, x_{1}\right)\right) \wedge \\
\text { concr }\left[\mathrm{m}_{5}\right] & =\mathrm{h}\left(\operatorname{pair}\left(\mathrm{n}_{2}, \mathrm{~b}\right) \wedge \operatorname{struct}\left[\mathrm{m}_{5}\right]=\mathrm{h}\left(\operatorname{pair}\left(\mathrm{n}_{2}, x_{2}\right)\right)\right.
\end{aligned}
$$

Here again $\beta$ respects the privacy of $\alpha$ (with respect to $\Sigma_{0}=\{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$ ) because we can find a model for each combination of values for $x_{1}, x_{2} \in\{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$. In contrast, if we had used the same nonce (replacing both $\mathrm{n}_{1}$ and $\mathrm{n}_{2}$ with n ), we would have that concr $\left[\mathrm{m}_{4}\right] \neq$ concr $\left[\mathrm{m}_{5}\right]$ and thus $x_{1} \neq x_{2}$, which does not follow from $\alpha$. Again the intruder does not find out $x_{1}$ or $x_{2}$ but only that the two users voted differently. The crucial point here (and the strength of ( $\alpha, \beta$ )-privacy) is that we do not have to specify checks for all the different things that the intruder may be able to figure out, or even think about them; we do not need to list all the different equivalences that should be considered. In ( $\alpha, \beta$ )-privacy, we simply just specify a formula $\alpha$ that describes what he is cleared to know and a formula $\beta$ containing all information that may be available to him.

The form of $\alpha$ and $\beta$ that we have used for Example 5 is at the core of many specifications, namely, when the intruder has observed a set of messages

[^7]and knows their structure. For this reason, we define a particular fragment of ( $\alpha, \beta$ )-privacy (for which we give some decidability results in Section 4) that deals only with what we call combinatoric $\alpha$ and only with the analysis of messages similar to the previous example.

Definition 8 (Combinatoric $\alpha$ ). We call $\alpha \in \mathcal{L}_{\Sigma_{0}}(\mathcal{V})$ combinatoric if $\Sigma_{0}$ is finite and contains only uninterpreted constants.

Thus, every model $\mathcal{I}$ of $\alpha$ maps the free variables of $\alpha$ to elements of the Herbrand universe induced by $\Sigma_{0}$. For each free variable $x$ of $\alpha$, we have $\mathcal{I}(x)=$ $[c] \approx$ for some unique $c \in \Sigma_{0}$. For every $\mathcal{I}$ such that $\mathcal{I} \models \alpha$, we define the substitution $\theta_{\mathcal{I}}$ that has as domain the set of free variables of $\alpha$, and such that $\theta_{\mathcal{I}}(x)=c$ iff $\mathcal{I}(x)=[c]_{\approx}$ (note that $\theta_{\mathcal{I}}$ is unique modulo $\left.\approx\right)$. Recall that, by slight abuse of notation, we may treat a substitution $\theta=\left[x_{1} \mapsto t_{1}, \ldots, x_{n} \mapsto t_{n}\right]$ as the Herbrand formula $x_{1}=t_{1} \wedge \ldots \wedge x_{n}=t_{n}$. Thus $\alpha$ is equivalent to the disjunction of all such substitutions:

Lemma 1. For every combinatoric $\alpha$, there is a finite set of substitutions $\Theta$ such that $\alpha \equiv \bigvee_{\theta \in \Theta} \theta$. We thus call $\Theta$ also the models of $\alpha$.

Proof. Let $\Theta=\left\{\theta_{\mathcal{I}} \mid \mathcal{I} \models \alpha\right\}$. Then $\Theta$ is finite since both the domain (namely the free variables of $\alpha$ ) and co-domain (namely $\Sigma_{0}$ ) of every $\theta_{\mathcal{I}}$ are finite. For every $\mathcal{I} \models \alpha$, it is clear that $\mathcal{I} \models \theta_{\mathcal{I}}$, thus $\alpha \models \bigvee_{\theta \in \Theta} \theta$. For every $\theta \in \Theta$ and every $\mathcal{I} \models \theta$, we have $\mathcal{I} \models \alpha$, since $\Sigma_{0}$ contains nothing but uninterpreted constants and all free variables of $\alpha$ are mapped to $\Sigma_{0}$ by $\theta$, thus also $\bigvee_{\theta \in \Theta} \theta \models$ $\alpha$.

Definition 9 (Message-analysis problem). Let $\alpha$ be combinatoric, $\theta$ a model of $\alpha$, struct $=\left\{\mathrm{m}_{1} \mapsto t_{1}, \ldots, \mathrm{~m}_{l} \mapsto t_{l}\right\}$ for some $t_{1}, \ldots, t_{l} \in \mathcal{T}_{\Sigma}(f v(\alpha))$, and concr $=\theta($ struct $)$. Define

$$
\operatorname{MsgAna}(\alpha, \text { struct }, \theta) \equiv \alpha \wedge \phi_{\text {frame }}(\text { concr }) \wedge \phi_{\text {frame }}(\text { struct }) \wedge \phi_{\text {concr } \sim \text { struct }}
$$

If $\beta \equiv \operatorname{MsgAna}(\alpha$, struct, $\theta)$, then we say that $\beta$ is a message-analysis problem (with respect to $\alpha$, struct, and $\theta$ ).

In general, such a $\beta$ allows us to model a system where messages $t_{i}$ have been exchanged that depend on some payload values $f v(\alpha)$ and the intruder has seen the concrete instantiations $\theta\left(t_{i}\right)$ of these messages. Typically, the intruder knowledge will contain all the values of $\Sigma_{0}$ but he does not know the substitution $\theta$, i.e., how the payload variables were actually chosen from $\Sigma_{0}$. What he knows, however, is the structure of the terms, i.e., where these variables occur in the $t_{i}$, because this structural information is usually part of a publicly available protocol description. He can try to exploit comparisons ( $\phi_{\text {concr~struct }}$ ) of concrete terms and structural information.

### 3.3 Pooling of Knowledge in ( $\alpha, \beta$ )-privacy

If we model several dishonest parties, an interesting question is what they can achieve if they collaborate, in particular, if they pool their knowledge. $(\alpha, \beta)$ privacy offers a particularly simple and declarative way of modeling, and reasoning about, the pooling of knowledge.

The general principle for pooling knowledge is as follows. We can describe the individual view of the world of two dishonest agents as two pairs $\left(\alpha_{1}, \beta_{1}\right)$ privacy and ( $\alpha_{2}, \beta_{2}$ )-privacy. Suppose they are message-analysis problems, i.e., $\beta_{1}=\operatorname{MsgAna}\left(\alpha_{1}\right.$, struct $\left._{1}, \theta_{1}\right)$ and $\beta_{2}=\operatorname{MsgAna}\left(\alpha_{2}\right.$, struct $\left._{2}, \theta_{2}\right)$, where we assume that the domains of struct ${ }_{1}$ and struct $_{2}$ are disjoint (i.e., we do not have a name clash in the memory locations).

Thus, $\alpha_{1}$ and $\alpha_{2}$ is the knowledge we have deliberately released to the two dishonest agents, and $\beta_{1}$ and $\beta_{2}$ is the technical information such as exchanged messages that they could observe, respectively. Of course, we require that both $\left(\alpha_{1}, \beta_{1}\right)$-privacy and ( $\alpha_{2}, \beta_{2}$ )-privacy already hold. If the two parties collude, then they can in general derive more than their individual $\alpha_{i}$, but it is quite "natural" to require that still ( $\alpha_{1} \wedge \alpha_{2}, \beta_{1} \wedge \beta_{2}$ )-privacy should hold. Why is it "natural"? It is obviously the strongest privacy requirement we can make: no system can prevent the intruders from combining their knowledge, i.e., both on the payload level $\alpha_{1} \wedge \alpha_{2}$ and on the technical level $\beta_{1} \wedge \beta_{2}$, and derive conclusions from that. The point is that the intruders should not be able to derive even more than that.

Example 6. Continuing in the realm of the previous examples, consider two dishonest agents who each have observed the messages around one vote $x_{1}$ and $x_{2}$, respectively:

$$
\begin{aligned}
\alpha_{1} & \equiv x_{1} \in\{\mathrm{a}, \mathrm{~b}, \mathrm{c}\} \\
\beta_{1} & \equiv \operatorname{MsgAna}\left(\alpha_{1}, \text { struct }_{1}, \theta_{1}\right)
\end{aligned}
$$

with struct ${ }_{1}=\left\{\mid \mathrm{m}_{1} \mapsto \mathrm{a}, \mathrm{m}_{2} \mapsto \mathrm{~b}, \mathrm{~m}_{3} \mapsto \mathrm{c}, \mathrm{m}_{4} \mapsto \mathrm{~h}\left(\operatorname{pair}\left(\mathrm{n}, x_{1}\right) \mid\right\}\right.$ and $\theta_{1}=\left\{x_{1} \mapsto\right.$ a\}, and

$$
\begin{aligned}
\alpha_{2} & \equiv x_{2} \in\{\mathrm{a}, \mathrm{~b}, \mathrm{c}\} \\
\beta_{2} & \equiv \operatorname{MsgAna}\left(\alpha_{2}, \text { struct }_{2}, \theta_{2}\right)
\end{aligned}
$$

with struct ${ }_{2}=\left\{\mid \mathrm{m}_{5} \mapsto \mathrm{a}, \mathrm{m}_{6} \mapsto \mathrm{~b}, \mathrm{~m}_{7} \mapsto \mathrm{c}, \mathrm{m}_{8} \mapsto \mathrm{~h}\left(\right.\right.$ pair $\left.\left(\mathrm{n}, x_{2}\right)\right\}$ and $\theta_{2}=\left\{x_{2} \mapsto\right.$ b\}.

Now consider $\beta_{1} \wedge \beta_{2}$. The question is whether it respects the privacy of $\alpha_{1} \wedge \alpha_{2}$. By joining forces and pooling their knowledge, and reasoning exactly like we did in Example 5, the two dishonest agents can derive that the two votes are different, i.e., $x_{1} \neq x_{2}$, which does not follow from $\alpha_{1} \wedge \alpha_{2}$. This is an example of the fact that, even though a protocol may safeguard privacy against two intruders who do not collude and see only part of the messages, the two intruders may be able to break the protocol's privacy when they pool their knowledge. As in the previous examples, the problem is that the two messages in question use the same nonce; if the hash-values in $\mathrm{m}_{4}$ and $\mathrm{m}_{8}$ used different nonces $\mathrm{n}_{1}$ and $\mathrm{n}_{2}$ (instead of n ), then the same derivation is not possible, and the privacy of $\alpha_{1} \wedge \alpha_{2}$ is respected.

Thus, in general, it makes sense to consider the case that dishonest agents can pool their knowledge and draw conclusions: no system can prevent this. From the $(\alpha, \beta)$-privacy paradigm it is clear that the conjunction of the respective $\alpha \mathrm{s}$ and the respective $\beta \mathrm{s}$ yields the best possible privacy requirement for the case of such a collusion, and this is a natural candidate to verify in a system.

### 3.4 Model-theoretical ( $\alpha, \beta$ )-privacy

We now introduce a model-theoretical view of $(\alpha, \beta)$-privacy that gives us additional tools to tackle some problems in a more semantical way. We then prove two theorems about the relationship between the classical notion of $(\alpha, \beta)$ privacy that we have considered above and the model-theoretical notion.

Recall that in Herbrand logic an interpretation does not specify a universe (as is the case in standard first-order logic), but the universe is rather the Herbrand universe induced by the considered alphabet (typically $\Sigma$ or $\Sigma_{0}$ in our case) and the congruence relation $\approx$. When it is not determined by the context, we may explicitly denote the alphabet and write, e.g., "a $\Sigma_{0}$-interpretation".

Definition 10 (Model-theoretical ( $\alpha, \beta$ )-privacy). Consider $\Sigma_{0}$ and $\Sigma$ as before, a formula $\alpha$ over $\Sigma_{0}$ and a formula $\beta$ over $\Sigma$ such that $f v(\alpha)=f v(\beta)$, both $\alpha$ and $\beta$ are consistent and $\beta=\alpha$. We say that $(\alpha, \beta)$-privacy holds modeltheoretically iff every $\Sigma_{0}$-model of $\alpha$ can be extended to a $\Sigma$-model of $\beta$, where a $\Sigma$-interpretation $\mathcal{I}^{\prime}$ is an extension of a $\Sigma_{0}$-interpretation $\mathcal{I}$ if they agree on all variables and all the interpreted function and relation symbols of $\Sigma_{0}$.
Theorem 1. If ( $\alpha, \beta$ )-privacy holds model-theoretically, then it also holds in the classical sense. Conversely, if for every model $\mathcal{I}$ of $\alpha$, there is a $\Sigma_{0}$-formula $\phi_{\mathcal{I}}$ that has only $\mathcal{I}$ as a model (with respect to $\Sigma_{0}$ ), then classical ( $\alpha, \beta$ )-privacy implies model-theoretical ( $\alpha, \beta$ )-privacy.

Proof. First, suppose that $(\alpha, \beta)$-privacy holds model-theoretically. Let $\mathcal{I}$ be a model of $\alpha$. Then, for some extension $\mathcal{I}^{\prime}$ of $\mathcal{I}$ (interpreting also the symbols in $\Sigma \backslash \Sigma_{0}$ ), we have $\mathcal{I}^{\prime} \models \beta$. Let further $\alpha^{\prime}$ be any $\Sigma_{0}$ formula that follows from $\beta$. Then, $\mathcal{I}^{\prime} \models \alpha^{\prime}$ and thus also $\mathcal{I} \models \alpha^{\prime}$. Since $\mathcal{I} \models \alpha$ was arbitrary, we have that $\alpha \models \alpha^{\prime}$; since $\alpha^{\prime}$ was arbitrary, we have classical $(\alpha, \beta)$-privacy.

Conversely, suppose that for every $\mathcal{I}$ such that $\mathcal{I} \models \alpha$ there is a $\Sigma_{0}$ formula $\phi_{\mathcal{I}}$ that has (with respect to $\Sigma_{0}$ ) exactly $\mathcal{I}$ as a model. Suppose, for the sake of contradiction, that $(\alpha, \beta)$-privacy does not hold in the model-theoretical sense, i.e., let $\mathcal{I} \models \alpha$ be such that no extension $\mathcal{I}^{\prime}$ to $\Sigma$ is a model of $\beta$. Thus, no model of $\beta$ is a model of $\phi_{\mathcal{I}}$, and then $\beta \models \neg \phi_{\mathcal{I}}$. Let thus $\alpha^{\prime} \equiv \phi_{\mathcal{I}}$. Obviously, $\alpha \not \vDash \alpha^{\prime}$, so classical ( $\alpha, \beta$ )-privacy is violated, and we conclude by contradiction.

Hence, for a combinatoric $\alpha$, classical $(\alpha, \beta)$-privacy and model-theoretical $(\alpha, \beta)$-privacy coincide. In general, however, classical $(\alpha, \beta)$-privacy does not imply model-theoretical ( $\alpha, \beta$ )-privacy.

Theorem 2. Classical ( $\alpha, \beta$ )-privacy does not imply model-theoretical ( $\alpha, \beta$ )privacy.

Proof. Let $\Sigma_{0}$ consist of the predicate $\mathrm{p} / 1$ and the uninterpreted functions $\mathrm{z} / 0$ and $s / 1$, i.e., the Herbrand Universe is the natural numbers and $p$ is a unary predicate over the natural numbers. We can represent any model of $p$ as an $\omega$-word $w \in\{0,1\}^{\omega}$, where the $i$-th position in the word is $1 \mathrm{iff} \mathrm{p}\left[\mathrm{s}^{i}(\mathrm{z})\right]$ holds. Let $\alpha \equiv$ true. Let now $\Sigma$ contain $\Sigma_{0}$ and the interpreted function $+/ 2$ and let
$\beta \equiv(\forall x, y \cdot(x+z=x) \wedge(x+\mathbf{s}(y)=\mathbf{s}(x+y))) \wedge\left(\exists t_{1}, t_{2} \cdot\left(t_{1} \neq \mathrm{z}\right) \wedge\left(\forall x . \mathrm{p}\left[t_{2}+x\right] \leftrightarrow \mathrm{p}\left[t_{1}+t_{2}+x\right]\right)\right)$
Thus, $\beta$ says that + is addition on natural numbers and that p when written as an $\omega$-word has the form $u v^{\omega}$ for some $u, v \in\{0,1\}^{*}, v \neq \epsilon$. Clearly, $(\alpha, \beta)$ privacy does not hold in the model-theoretical sense, but we now show that it holds in the classical sense.

Let $\alpha^{\prime}$ be any formula over $\Sigma_{0}$, and let $G$ be the models for p of the formula $\neg \alpha^{\prime}$ represented as $\omega$-words. Now $G$ is an $\omega$-regular language: this is because $\neg \alpha^{\prime}$ is over $\Sigma_{0}$, thus can only use $\mathrm{z}, \mathrm{s}$ and p besides the symbols of the logic; the meaning of $z$ and $s$ and the domain of $p$ are determined by the Herbrand universe, thus $\neg \alpha^{\prime}$ corresponds to an S1S formula and the set of its models forms an $\omega$-regular language [11]. We can distinguish two cases. In the first case, $G=\emptyset$, then $\alpha=\alpha^{\prime}$, so $(\alpha, \beta)$-privacy is not violated for this $\alpha^{\prime}$. In the second case, $G \neq \emptyset$. A non-empty $\omega$-regular language, however, always contains a word of the form $w=u v^{\omega}$ for some $u, v \in\{0,1\}^{*}, v \neq \epsilon$. Then, $w$ can be extended to a model of $\beta$, but it is also a model of $\neg \alpha^{\prime}$, and thus $\beta \not \vDash \alpha^{\prime}$. Hence, also for this $\alpha^{\prime},(\alpha, \beta)$-privacy is not violated either. In conclusion, there is no $\alpha^{\prime}$ over $\Sigma_{0}$ such that $\beta \models \alpha^{\prime}$ but $\alpha \not \vDash \alpha^{\prime}$.

Note again that whenever all models of $\alpha$ can be written as $\Sigma_{0}$-formulaewhich holds in all but quite construed cases-classical and model-theoretical $(\alpha, \beta)$-privacy holds. This allows us to overcome the following obstacle. $(\alpha, \beta)$ privacy asks for any $\Sigma_{0}$-formula $\alpha^{\prime}$ that can be derived from $\beta$ but not from $\alpha$. In general, there is a (countably) infinite choice for $\alpha^{\prime}$ to consider. However, this is not the case when $\alpha$ is combinatoric:

Corollary 1. Consider an $(\alpha, \beta)$ pair, where $\alpha$ is combinatoric and $\Theta$ is the set of models of $\alpha$. Then, $\beta$ violates the privacy of $\alpha$ iff $\beta \models \neg \theta$ for any $\theta \in \Theta$.

Proof. For a combinatoric $\alpha$, classical and model-theoretical privacy coincide by Theorem 1. Moreover, for a combinatoric $\alpha, \Theta$ is finite by definition. The statement follows by the fact that $\beta \models \neg \theta$ is equivalent to $\neg \operatorname{Sat}(\beta \wedge \theta)$.

## 4 Automation and the Relation to Static Equivalence

The concept of ( $\alpha, \beta$ )-privacy is very expressive, because Herbrand logic is. Considering Example 2, we recall that we can axiomatize arithmetic (of natural numbers) by a Herbrand formula $\alpha$ so that $\alpha \models \gamma$ iff $\gamma$ is a true sentence of arithmetic. Let valid be a further nullary relation symbol in $\Sigma_{0}$ and $\beta \equiv$
$\alpha \wedge(\gamma \Longrightarrow$ valid $)$; then $\beta$ respects the privacy of $\alpha$ iff $\gamma$ is a true sentence of arithmetic. Thus, in general, $(\alpha, \beta)$-privacy (or its complement) is not even semi-decidable.

We see this expressive power as a feature, because it allows us to think about privacy without the tight corset imposed by automated methods. In this section, we explore a decidable fragment and the relation to static equivalence of frames for which many decidability results already exist. Because of its expressive power, it is no surprise that one can formulate static equivalence in Herbrand logic, and thus reduce static equivalence of frames to $(\alpha, \beta)$-privacy.
Lemma 2. Given two frames $\digamma_{1}$ and $\digamma_{2}$, we have $\digamma_{1} \sim \digamma_{2}$ iff $(\alpha, \beta)$-privacy holds for $\alpha \equiv x \in\{0,1\}$ and $\beta \equiv \alpha \wedge\left(x=1 \Longrightarrow \phi_{\text {frame }}\left(\digamma_{1}\right) \wedge \phi_{\text {frame }}\left(\digamma_{2}\right) \wedge\right.$ $\left.\phi_{\digamma_{1} \sim \digamma_{2}}\right)$.

Proof. The statement follows straightforwardly by observing that every model of $\alpha$ can be extended to a model of $\phi_{\text {frame }}\left(\digamma_{1}\right) \wedge \phi_{\text {frame }}\left(\digamma_{2}\right) \wedge \phi_{\digamma_{1} \sim \digamma_{2}}$ iff $\digamma_{1} \sim \digamma_{2}$.

The simple argument of this lemma may seem slightly unfair towards static equivalence of frames, since we are not truly using $\alpha$ for the high-level payload information available to the intruder, but rather considering everything as technical, and then just exploiting the expressive power of Herbrand logic. In order to show a closer relationship between static equivalence and $(\alpha, \beta)$ privacy, we prove below that a large fragment of the static equivalence problem for frames can be encoded into the message-analysis fragment of $(\alpha, \beta)$-privacy (cf. Definition 9).

Static equivalence of frames is essentially the question whether the intruder can distinguish two concrete worlds. For instance, the frames $\digamma_{1}$ and $\digamma_{2}$ in Example 3 and Example 4 represent two concrete worlds that the intruder can distinguish: $\digamma_{1} \not \nsim \digamma_{2}$. In contrast, $(\alpha, \beta)$-privacy expresses with $\alpha$ all possible worlds (there may be more than two) and with $\beta$ one concrete world, asking whether the intruder can exclude some of the worlds that are models of $\alpha$. This, in particular, requires a distinction-that frames do not have-between high-level payload information and low-level technical information.

The fact that static equivalence problems can be somehow encoded into ( $\alpha, \beta$ )-privacy is not too surprising since it simply exploits the expressiveness of Herbrand logic, as we already remarked. A deeper relation is highlighted in the following where we show that message-analysis-style problems with a clear distinction of payload and technical information can be "more directly" encoded into ( $\alpha, \beta$ )-privacy.

The other direction for message-analysis problems is also possible. This is interesting and useful not just conceptually but also practically, since it allows us to use for $(\alpha, \beta)$-privacy existing results and tools for static equivalence (in particular, for various algebraic theories).

As first step towards a relation between static equivalence and ( $\alpha, \beta$ )-privacy, we show the following:

Lemma 3. Let $\alpha$ be combinatoric, $\Theta$ be the models of $\alpha$, and $\beta \equiv \operatorname{MsgAna}\left(\alpha, \digamma, \theta_{1}\right)$ for some $\theta_{1} \in \Theta$. Then, for every $\theta_{2} \in \Theta$, we have Sat $\left(\theta_{2} \wedge \beta\right)$ iff $\theta_{1}(\digamma) \sim \theta_{2}(\digamma)$.

Proof. The following statements are equivalent:

- $\operatorname{Sat}\left(\theta_{2} \wedge \beta\right)$
- $\operatorname{Sat}\left(\theta_{2} \wedge \alpha \wedge \phi_{\text {frame }}\left(\theta_{1}(\digamma)\right) \wedge \phi_{\text {frame }}(\digamma) \wedge \phi_{\theta_{1}(\digamma) \sim \digamma}\right)$ [by Definition 9]
- $\operatorname{Sat}\left(\theta_{2} \wedge \alpha \wedge \phi_{\text {frame }}\left(\theta_{1}(\digamma)\right) \wedge \phi_{\text {frame }}\left(\theta_{2}(\digamma)\right) \wedge \phi_{\theta_{1}(\digamma) \sim \theta_{2}(\digamma)}\right)$

This is because the formula contains $\theta_{2}$ and thus all variables must be instantiated accordingly, including those occurring in $\digamma$.

- $\operatorname{Sat}\left(\phi_{\text {frame }}\left(\theta_{1}(\digamma)\right) \wedge \phi_{\text {frame }}\left(\theta_{2}(\digamma)\right) \wedge \phi_{\theta_{1}(\digamma) \sim \theta_{2}(\digamma)}\right)$

This is because the formula $\operatorname{Sat}\left(\phi_{\text {frame }}\left(\theta_{1}(\digamma)\right) \wedge \phi_{\text {frame }}\left(\theta_{2}(\digamma)\right) \wedge \phi_{\theta_{1}(\digamma) \sim \theta_{2}(\digamma)}\right)$ is ground (and consistent with $\alpha$ and $\theta_{2}$ ).

- $\theta_{1}(\digamma) \sim \theta_{2}(\digamma)$ [by Definition 6]

We now show the equivalence of message-analysis problems with a corresponding finite set of static equivalence problems. More specifically, we prove that it suffices to pick arbitrarily one of the models of $\alpha$ and show that ( $\alpha, \beta$ )privacy holds for that model, and that this is equivalent to showing the indistinguishability of the given frame under every model of $\alpha$.
Theorem 3. Let $\alpha$ be combinatoric, $\Theta=\left\{\theta_{1}, \ldots, \theta_{n}\right\}$ be the models of $\alpha$, and $\beta \equiv \operatorname{MsgAna}\left(\alpha, \digamma, \theta_{1}\right)$ for some $\theta_{1} \in \Theta$. Then, we have that $(\alpha, \beta)$-privacy holds iff $\theta_{1}(\digamma) \sim \ldots \sim \theta_{n}(\digamma)$.

Proof. The following are equivalent:

- $(\alpha, \beta)$-privacy
- $\operatorname{Sat}\left(\theta_{i} \wedge \beta\right)$ for every $1 \leq i \leq n$ [by Definition 10]
- $\theta_{i}(\digamma) \sim \theta_{1}(\digamma)$ for every $1 \leq i \leq n$ [by Lemma 3]
- $\theta_{1}(\digamma) \sim \ldots \sim \theta_{n}(\digamma)$ [by Definition of $\sim$ ]

Example 7. Let us consider again the second case of our simple voting example (Example 5), in which the voting server publishes messages of the form $\mathrm{h}\left(\operatorname{pair}\left(\mathrm{n}, x_{i}\right)\right)$ for a fixed number n known only to the server, i.e., n is a secret from $\Sigma \backslash \Sigma_{0}$. The models of

$$
\alpha \equiv x \in\{\mathrm{a}, \mathrm{~b}, \mathrm{c}\}
$$

are $\Theta=\left\{\theta_{1}, \theta_{2}, \theta_{3}\right\}$ with $\theta_{1}=\{x \mapsto \mathrm{a}\}, \theta_{2}=\{x \mapsto \mathrm{~b}\}$ and $\theta_{3}=\{x \mapsto \mathrm{c}\}$. Then, for $\theta_{i} \in \Theta$ corresponding to the specific vote $x_{i}$ that has been cast, we have

$$
\begin{aligned}
\beta & \equiv M \operatorname{sgAna}\left(\alpha, \text { struct }, \theta_{i}\right) \\
& \equiv \alpha \wedge \phi_{\text {frame }}(\text { concr }) \wedge \phi_{\text {frame }}(\text { struct }) \wedge \phi_{\text {concr } \sim \text { struct }}
\end{aligned}
$$

for a frame struct $=\left\{\left|\mathrm{m}_{1} \mapsto \mathrm{a}, \mathrm{m}_{2} \mapsto \mathrm{~b}, \mathrm{~m}_{3} \mapsto \mathrm{c}, \mathrm{m}_{4} \mapsto \mathrm{~h}(\operatorname{pair}(\mathrm{n}, x))\right|\right\}$, with concr $=\theta_{i}($ struct $)$. So, to check whether ( $\alpha, \beta$ )-privacy holds it suffices to check one $\theta_{i} \in \Theta$, say $\theta_{1}=\{x \mapsto a\}$, and then it holds for all $\theta_{i} \in \Theta$. Theorem 3 tells us that this is equivalent to checking the static equivalence between all concrete votes by considering all models, i.e., $\theta_{1}($ struct $) \sim \theta_{2}($ struct $) \sim \theta_{3}($ struct $)$.

Corollary 2. Let $\alpha$ be combinatoric, $\Theta$ be the models of $\alpha$, and $\digamma$ be a frame. Then, we have that ( $\alpha, \operatorname{MsgAna}(\alpha, \digamma, \theta)$ )-privacy holds for some $\theta \in \Theta$ iff $(\alpha, M \operatorname{sgAna}(\alpha, \digamma, \theta))$-privacy holds for all $\theta \in \Theta$.

Hence, if in a given world $\theta$ the intruder cannot exclude any other worlds, then in no world he can exclude any world. Thus, in transition systems, we may summarize the worlds that only differ on the concrete value of the privacy variables in one state. This is however only possible as long as no behavior of the honest agents depends on it. For instance, in a voting protocol, the actual values of the vote have no influence on the behavior of the honest agents, so there we can always model all possible voting outcomes by a single state that is parameterized over the choice of a model $\theta$. Thus, we can exploit privacy (the goal) even for an efficient representation.

Since this result is independent of the considered set $\Sigma_{o p}$ of cryptographic operations and algebraic theory, we immediately have that if we can decide static equivalence for a given theory (e.g., $[3,10]$ ), then we can decide the message-analysis problem fragment of $(\alpha, \beta)$-privacy for that theory.

To conclude the section, note that, instead of relying on static equivalence, we could have also given a direct decision procedure for our example theory, without an enumeration of all models.

## 5 Modeling and Reasoning About Further Example Scenarios

We chose the following major areas to model further examples and show $(\alpha, \beta)$ privacy at work:

- randomized vs. non-randomized encryption including non-determinism and the notion of strong secrecy (Section 5.1),
- guessing attacks (Section 5.2, in which we discuss different approaches to encode passwords and guessing in ( $\alpha, \beta$ )-privacy and show unique features of our logic), and
- e-voting (Section 5.3).


### 5.1 Modeling Subtleties of Encryption

### 5.1.1 Randomized vs. Non-Randomized Encryption

Standard Dolev-Yao models are blind to the problem of non-randomized encryption because the intruder cannot compare results in these models. Here, we
have directly modeled randomized encryption operators by means of a random value $r$, e.g., $\operatorname{crypt}(k, r, m)$, where the randomization should, of course, be different at each encryption. In our approach, this is taken care of by the transition system, which should actually ensure that every honest agent chooses a new randomness at each encryption.

We model non-randomized encryption by using $r=\varepsilon$ for an intruder-known constant $\varepsilon$, i.e., we write $\operatorname{crypt}(\cdot, \varepsilon, \cdot)$ to model the missing randomization.

Example 8 (Simple Voting). Let $\Sigma_{0}=\{0,1\}, \alpha \equiv x \in\{0,1\}$, and $\beta \equiv$ $\operatorname{MsgAna}(\alpha$, struct, $\theta)$, where struct $=\left\{\mid \mathrm{m}_{1} \mapsto \varepsilon, \mathrm{~m}_{2} \mapsto 0, \mathrm{~m}_{3} \mapsto 1, \mathrm{~m}_{4} \mapsto\right.$ $\left.\mathrm{k}, \mathrm{m}_{5} \mapsto \operatorname{crypt}(\mathrm{k}, \varepsilon, x)\right\}$ and $\theta=\{x \mapsto 0\}$. The intruder can derive from $\beta$ that concr $\left[\mathrm{m}_{5}\right]=\operatorname{crypt}(\mathrm{k}, \varepsilon, 0)=\operatorname{concr}\left[\operatorname{crypt}\left(\mathrm{m}_{4}, \mathrm{~m}_{1}, \mathrm{~m}_{2}\right)\right]$, and then, by $\phi_{\digamma_{1} \sim \digamma_{2}}$, $\operatorname{struct}\left[\mathrm{m}_{5}\right]=\operatorname{struct}\left[\operatorname{crypt}\left(\mathrm{m}_{4}, \mathrm{~m}_{1}, \mathrm{~m}_{2}\right)\right]$ and thus $x=0$.

A similar deduction would not be possible if crypt contained additionally something random, i.e., if struct $=\left\{\mathrm{m}_{1} \mapsto \varepsilon, \mathrm{~m}_{2} \mapsto 0, \mathrm{~m}_{3} \mapsto 1, \mathrm{~m}_{4} \mapsto \mathrm{k}, \mathrm{m}_{5} \mapsto\right.$ $\operatorname{crypt}(\mathrm{k}, \mathrm{r}, x) \mid\}$, where r is not known to the intruder. (This is of course the very reason for probabilistic encryption.) In such a probabilistic variant, $\beta$ would indeed respect the privacy of $\alpha$ as the intruder would no longer be able to generate terms that would give him any interesting insight on $x$.

### 5.1.2 Non-determinism

Another interesting question that we can ask is what the intruder can discover when he does not know the content of messages sent by the honest agents (in fact, he might not even know to which protocol the messages belong), but he knows the format of the messages that are exchanged in several protocols. $(\alpha, \beta)$ privacy can easily encode non-determinism, e.g., if messages have a different structure depending on the choices of parties. Consider this example

$$
\alpha \equiv x \in\{\text { yes }, \mathrm{no}\} \wedge y \in\{\mathrm{a}, \mathrm{~b}\}
$$

where $x$ is the server's decision and $y$ is the concrete name of the client (the intruder knows yes, no and pub(a)). A party may respond to a request with

$$
\operatorname{crypt}(\operatorname{pub}(y), r, \operatorname{pair}(y e s, n)) \quad \text { or } \quad \operatorname{crypt}(\operatorname{pub}(y), r, n o),
$$

depending on the decision on the vote, where n is a secret from $\Sigma \backslash \Sigma_{0}$ (and where, like in Example 5, we abstract away from the fact that the intruder might be able to guess such a fixed number). This is no longer a message-analysis problem, since the structure of the messages depends on $x$; and it cannot be directly expressed as a problem of static equivalence of frames either. ${ }^{8}$ However, $(\alpha, \beta)$-privacy allows us to incorporate such more complex relations between $\alpha$ and the structure of messages, e.g., the history of events that happened so far. For this example, we can specify the struct and concr knowledge of the intruder

[^8]directly as a formula related to the flag $x$ as part of $\beta$ :

```
\(\operatorname{concr}\left[\mathrm{m}_{1}\right]=\operatorname{struct}\left[\mathrm{m}_{1}\right]=\) yes \(\wedge\)
\(\operatorname{concr}\left[\mathrm{m}_{2}\right]=\operatorname{struct}\left[\mathrm{m}_{2}\right]=\) no \(\wedge\)
concr \(\left[\mathrm{m}_{3}\right]=\operatorname{struct}\left[\mathrm{m}_{3}\right]=\operatorname{pub}(\mathrm{a}) \wedge\)
concr \(\left[\mathrm{m}_{4}\right]=\operatorname{crypt}(\operatorname{pub}(\mathrm{a}), \mathrm{r}, \operatorname{pair}(\mathrm{yes}, \mathrm{n})) \wedge\)
    \(\left(\left(x=\right.\right.\) yes \(\wedge \operatorname{struct}\left[\mathrm{m}_{4}\right]=\operatorname{crypt}(\operatorname{pub}(y), \mathrm{r}, \operatorname{pair}(\) yes, n\(\left.))\right) \vee\)
    \(\left.\left(x=\operatorname{no} \wedge \operatorname{struct}\left[\mathrm{m}_{4}\right]=\operatorname{crypt}(\operatorname{pub}(y), \mathrm{r}, \mathrm{no})\right)\right)\)
```

so that $\beta$ actually encodes a concrete world (in this case, such that $x=$ yes).
Indeed, in the example, $(\alpha, \beta)$-privacy holds, but if we had used non-randomized encryption (replacing $r$ with $\varepsilon$ ), the intruder could have generated the term $\operatorname{crypt}\left(\mathrm{m}_{3}, \varepsilon, \mathrm{~m}_{2}\right)$ and checked that it does not produce the same concr value as $\mathrm{m}_{4}$. Thus, $\beta \models \operatorname{struct}\left[\mathrm{m}_{4}\right] \neq \operatorname{crypt}(\operatorname{pub}(\mathrm{a}), \varepsilon, \mathrm{no})$. It follows that the model $(x=$ no $\wedge y=\mathrm{a})$ is excluded. Note that this does not tell us the value of $x$ nor that of $y$, but just excludes one of the combinations.

### 5.1.3 Strong Secrecy

In the static equivalence community (e.g., [18]), there is also the notion of strong secrecy:

Definition 11. A frame $\digamma$ that talks about a variable $x$ (the "secret") respects the strong secrecy of $x$ if $\digamma\{x \mapsto s\} \sim \digamma\{x \mapsto t\}$ for any generable terms $s$ and $t$.

Thus, if the intruder can choose arbitrary (known) values and the secret is replaced for those values, he still cannot deduce which one it is.

The philosophy of the ( $\alpha, \beta$ )-privacy approach is actually a bit different from this view: we would normally consider problems of strong secrecy actually as the question of checking that a protocol provides secrecy even when the secrets are weak, like poor passwords. We illustrate this in the following subsection on modeling guessing attacks. However, the notion of a game that the intruder can play is quite interesting for its relation for instance to cryptographic models. Therefore, we quickly illustrate how this can be done with ( $\alpha, \beta$ )-privacy.

Example 9 (Strong Secrecy Game). As an example of the experiment that the intruder can do with his environment, consider the following "strong secrecy game". To model an interaction between the intruder and a (virtual) host, we design a formula that represents the evolution of the intruder knowledge in the game in two steps:

- We start in a state where the intruder knowledge is represented by $\digamma_{1}=$ $\left\{\left|\mathrm{m}_{1} \mapsto \mathrm{n}_{1}, \mathrm{~m}_{2} \mapsto \mathrm{n}_{2}\right|\right\}$.
- The intruder can choose any two recipes $s_{0}$ and $s_{1}$ with respect to $\digamma_{1}$, i.e., $\operatorname{gen}_{\digamma_{1}}\left(s_{0}\right) \wedge g e n_{\digamma_{1}}\left(s_{1}\right)$, and he sends concr $\left[s_{0}\right]$ and concr $\left[s_{1}\right]$ to an honest agent host.
- Now the host non-deterministically chooses a Boolean $b \in\{0,1\}$ and encrypts concr $\left[s_{b}\right]$ with a key k and sends it back to the intruder, who only knows that the encryption contains $s_{b}$ depending on the choice $b$ (that the intruder does not know). For this uncertainty, we introduce a new variable $x$ and have $\alpha \equiv x \in\{0,1\}$. Let $\theta=[x \mapsto b]$. Now we define

$$
\begin{aligned}
& \beta \equiv \operatorname{MsgAna}\left(\alpha, \digamma_{1}, \theta\right) \\
& \wedge \\
& \exists s_{0}, s_{1} \cdot g e n_{\digamma_{1}}\left(s_{0}\right) \wedge \operatorname{gen}_{\digamma_{1}}\left(s_{1}\right) \\
& \wedge \exists s_{x} \cdot s_{x} \in\left\{s_{0}, s_{1}\right\} . \\
& \quad\left(\left(x=0 \wedge \operatorname{MgAAna}\left(\alpha, \digamma_{2}, \theta \cup\left[s_{x} \mapsto s_{0}\right]\right)\right) \vee\right. \\
&\left.\quad\left(x=1 \wedge \operatorname{MsgAna}\left(\alpha, \digamma_{2}, \theta \cup\left[s_{x} \mapsto s_{1}\right]\right)\right)\right)
\end{aligned}
$$

where $\digamma_{2}=\digamma_{1} \cup\left\{\mathrm{~m}_{3} \mapsto \operatorname{scrypt}\left(k\right.\right.$, concr $\left.\left.\left[s_{x}\right]\right)\right\}$.
Indeed, ( $\alpha, \beta$ )-privacy holds, i.e., the intruder cannot determine $x$, so this gives us strong secrecy. An example of violation of strong secrecy would be for instance if the message had rather the form $\operatorname{scrypt}\left(\operatorname{concr}\left[s_{b}\right], \ldots\right.$ ) (i.e., if a weak secret is used to encrypt a message) or if we have hash-values like $h\left(\operatorname{concr}\left[s_{b}\right], \ldots\right)$ with other guessable elements, since then the intruder can verify whether the message was generated with $s_{0}$ or $s_{1}$, and thus obtain $x$.

### 5.2 Modeling Guessing Attacks

Guessing attacks have been intensively studied in security protocol analysis, also using static equivalence, e.g., $[2,26,30]$ to name just a few works. We can use also this example to show how flexible our ( $\alpha, \beta$ )-privacy approach is: suppose that we have an intruder who knows only a part of the password space (e.g., some users use good passwords, some use bad passwords). Let $P=\left\{\mathrm{p}_{1}, \ldots, \mathrm{p}_{k}\right\}$ be the space of all passwords and let $D=\left\{\mathrm{p}_{1}, \ldots, \mathrm{p}_{l}\right\}$, with $l<k$, be the intruder's dictionary. For concreteness, let us consider the classical example of the MS CHAPv2 protocol [40], where the server sends a nonce ns and the client c should produce $\mathrm{nc}, \mathrm{h}_{x}(\mathrm{nc}, \mathrm{ns}, \mathrm{c})$ for some client nonce nc , hash-MAC'ed with the client's password $x$. Then one reachable state could be:

$$
\begin{aligned}
\alpha & \equiv x \in\left\{\mathrm{p}_{1}, \ldots, \mathrm{p}_{k}\right\} \\
\beta & \equiv \operatorname{MsgAna(\alpha ,\text {struct},\theta )}
\end{aligned}
$$

where struct $=\left\{\mid \mathrm{m}_{1} \mapsto \mathrm{p}_{1}, \mathrm{~m}_{2} \mapsto \mathrm{p}_{2}, \ldots, \mathrm{~m}_{l} \mapsto \mathrm{p}_{l}, \mathrm{~m}_{l+1} \mapsto \mathrm{~ns}, \mathrm{~m}_{l+2} \mapsto \mathrm{c}, \mathrm{m}_{l+3} \mapsto\right.$ $\operatorname{pair}(\mathrm{nc}, \mathrm{h}(x, \operatorname{pair}(\mathrm{nc}, \operatorname{pair}(\mathrm{ns}, \mathrm{c}))))\}$ and $\theta=\left\{x \mapsto \mathrm{p}_{1}\right\}$.

This is obviously the concrete state where $x=\mathrm{p}_{1}$, and the intruder can find out that that is indeed the case. To that end, he uses the generable terms $s=\mathrm{h}\left(\mathrm{m}_{1}, \operatorname{pair}\left(\operatorname{proj}_{1}\left(\mathrm{~m}_{l+3}\right)\right.\right.$, $\left.\left.\operatorname{pair}\left(\mathrm{m}_{l+1}, \mathrm{~m}_{l+2}\right)\right)\right)$ and $t=\operatorname{proj}_{2}\left(\mathrm{~m}_{l+3}\right)$, so that concr $[s]=$ concr $[t]$ and thus struct $[s]=\operatorname{struct}[t]$. Then

$$
\begin{aligned}
& \mathrm{h}\left(\mathrm{p}_{1}, \operatorname{pair}\left(\operatorname{proj}_{1}(\operatorname{pair}(\mathrm{nc}, \mathrm{~h}(x, \operatorname{pair}(\mathrm{nc}, \operatorname{pair}(\mathrm{~ns}, \mathrm{c}))))), \operatorname{pair}(\mathrm{ns}, \mathrm{c})\right)\right)= \\
& \operatorname{proj}_{2}(\operatorname{pair}(\mathrm{nc}, \mathrm{~h}(x, \operatorname{pair}(\mathrm{nc}, \operatorname{pair}(\mathrm{~ns}, \mathrm{c})))))
\end{aligned}
$$

and thus $\mathrm{h}\left(\mathrm{p}_{1}, \operatorname{pair}(\mathrm{nc}, \operatorname{pair}(\mathrm{ns}, \mathrm{c}))\right)=\mathrm{h}(x, \operatorname{pair}(\mathrm{nc}, \operatorname{pair}(\mathrm{ns}, \mathrm{c})))$. In the given Herbrand universe (which depends on the algebraic theory), the only possible model for $x$ is $x=\mathrm{p}_{1}$, thus $\beta \models x=\mathrm{p}_{1}$.

Note that if we consider a trace where $x=\mathrm{p}_{n}$ with $l<n \leq k$, thus using a password that the intruder does not have in his dictionary, then this attack is not possible. However, the intruder can still derive $x \in\left\{\mathbf{p}_{l+1}, \ldots, \mathbf{p}_{k}\right\}$ since by checking his entire dictionary, he can confirm that the password is not one of those he knows. So, in fact, privacy is violated also in these cases, even though most people would agree that this in itself is not a problem, at least for online guessing: an interesting argument, in this case, is that even the best protocol and the best password cannot prevent the intruder from trying out online a few passwords and confirming that they are wrong guesses. Whereas in offline guessing good password protocols don't in fact allow the intruder to carry out offline guesses, we can still consider transition systems in which the intruder can actually send guesses to the server. We leave a more detailed investigation of this for future work.

In fact, for this reason we suggest to see the password itself as a technical information that encrypts the actual payload information we try to protect. This is a simple way to say: we don't care what the intruder finds out about the password, but about the messages that are encrypted with it. This allows us to circumvent all troubles with declassification of information or quantitative aspects. Here is how this could look like, if the password $p_{n}$ is used to encrypt a payload $x$ that is, say, from a set of values $\left\{\mathrm{c}_{1}, \ldots, \mathrm{c}_{m}\right\}$ :

$$
\begin{aligned}
\alpha & \equiv x \in\left\{\mathrm{c}_{1}, \ldots, \mathrm{c}_{m}\right\} \\
\beta & \equiv \operatorname{MsgAna(\alpha ,\text {struct},\theta )}
\end{aligned}
$$

where struct $=\left\{\mid \mathrm{m}_{1} \mapsto \mathrm{p}_{1}, \mathrm{~m}_{2} \mapsto \mathrm{p}_{2}, \ldots, \mathrm{~m}_{l} \mapsto \mathrm{p}_{l}, \mathrm{~m}_{l+1} \mapsto \mathrm{~ns}, \mathrm{~m}_{l+2} \mapsto \mathrm{c}, \mathrm{m}_{l+3} \mapsto\right.$ $\left.\operatorname{pair}\left(\mathrm{nc}, \mathrm{h}\left(\mathrm{p}_{n}, \operatorname{pair}(\mathrm{nc}, \operatorname{pair}(\mathrm{ns}, \mathrm{c}))\right)\right), \mathrm{m}_{l+4} \mapsto \operatorname{scrypt}\left(\mathrm{p}_{n}, x\right)\right\}$ and $\theta=\left\{x \mapsto \mathrm{c}_{7}\right\}$.

This preserves privacy if $l<n \leq k$ (i.e., for a good password) and still violates it for a bad password $1 \leq n \leq l$. Note that when the password is technical information, we do no longer use a variable for it, because it is not part of the space we try to get information about.

### 5.3 Modeling E-Voting

An interesting field for privacy goals is of course electronic voting (e-voting), which we already touched upon in the previous examples. In this paper, we do not want to model and analyze a full voting protocol, as that is not the main focus of our research, but rather only illustrate how ( $\alpha, \beta$ )-privacy gives a new and declarative way to formulate privacy goals and describe their analysis.

Let us first look at the most basic setting: the vote between two options 1 and 0 (e.g., representing "yes" or "no", or the choice between two candidates).
Definition 12 (Binary Voting Privacy). Consider a voting system where the choice is either 1 or 0 . Let $N$ be the number of cast votes and $R$ be the number
of votes for 1 . Let $\Sigma_{0}$ consist of the constant 0 , the successor function $s(\cdot)$ and the addition function + , and consider the axiom $\alpha_{a x}$ characterizing addition:

$$
\alpha_{a x} \equiv \forall x . \forall y . x+0=x \wedge x+s(y)=s(x+y)
$$

The privacy goal for binary votes (namely releasing $N$ and $R$ ) is then defined by the following formula $\alpha$ :

$$
\alpha \equiv \alpha_{a x} \wedge v_{1} \in\{0,1\} \wedge \ldots \wedge v_{N} \in\{0,1\} \wedge v_{1}+v_{2}+\ldots+v_{N}=R
$$

where $v_{1}, \ldots, v_{N}$ are variables.
Note that, strictly speaking, $\alpha$ is not combinatoric, since $\Sigma_{0}$ contains 0 and $s$ (so the Herbrand universe entails the natural numbers) and + is interpreted as addition. There are however only finitely many models (with respect to the free variables). In fact, there exists an (inefficient) encoding into a combinatoric problem that works without a concept of natural numbers.

Example 10. For example, consider the trivial voting protocol, in which every voter $i$ sends their vote $v_{i}$ directly to a voting server, signed with their own private key and encrypted with the server's public key. This of course makes the strong requirement that the server must be completely trusted by everybody as it can see all votes and nobody can verify the correct tallying by the server. Let $\theta$ be the true result of the vote (mapping each $v_{i}$ to 0 or 1 ); then we have $\beta=\operatorname{MsgAna}(\alpha$, struct, $\theta)$ where

$$
\left.\left.\begin{array}{rl}
\text { struct }=\{ & m_{1} \mapsto \operatorname{crypt}\left(\operatorname{pub}(s), r_{1}, \operatorname{sign}\left(\operatorname{priv}(1), \operatorname{ballot}\left(v_{1}\right)\right)\right), \\
& \ldots \\
& m_{N}
\end{array}>\operatorname{crypt}\left(\operatorname{pub}(s), r_{N}, \operatorname{sign}\left(\operatorname{priv}(N), \operatorname{ballot}\left(v_{N}\right)\right)\right)\right\}\right\}
$$

and ballot is the ballot format of this vote, $\operatorname{priv}(i)$ is the private key of the ith voter, $\operatorname{pub}(s)$ is the public key of the server, and the $r_{i}$ are random values to make the encryption non-deterministic.

In this example, ( $\alpha, \beta$ )-privacy holds, since the intruder cannot analyze any messages and cannot compare them, thus every model of $\alpha$ can be extended to a model of $\beta$.

There are many works on electronic voting using the applied pi-calculus or a similar calculus to model the protocol (see, e.g., $[21,7,14,15,17,19]$ to name a few). The voting privacy goal is usually then expressed by the following trick: consider two processes that differ only in the concrete vote of two (honest) voters who swap their vote; then any such pair of processes must be indistinguishable. More formally, let $\theta_{1}$ and $\theta_{2}$ be two substitutions with domain $v_{1}, \ldots, v_{N}$ and codomain $\{0,1\}$. Then $\theta_{1}$ and $\theta_{2}$ are a vote swap of each other, if $\theta_{1}\left(v_{i}\right)=\theta_{2}\left(v_{j}\right)$ and $\theta_{1}\left(v_{j}\right)=\theta_{2}\left(v_{i}\right)$ for some $i, j \in\{1, \ldots, N\}$ and $\theta_{1}\left(v_{k}\right)=\theta_{2}\left(v_{k}\right)$ for all other $k$. We show that this privacy definition based on vote swapping is actually in some sense equivalent to the binary voting privacy goal:

Theorem 4. Let $\alpha$ be the binary voting privacy goal from Definition 12, $\theta_{0}$ be a model of $\alpha$, and $\beta \equiv \operatorname{MsgAna}\left(\alpha\right.$, struct, $\left.\theta_{0}\right)$ for some frame struct be a message-analysis problem. Then ( $\alpha, \beta$ )-privacy holds iff $\theta_{1}($ struct $) \sim \theta_{2}($ struct $)$ for any two models $\theta_{1}$ and $\theta_{2}$ of $\alpha$ that are a vote swap of each other.

Proof. Let $\Theta=\left\{\theta_{1}, \ldots, \theta_{n}\right\}$ be the models of $\alpha$. By Theorem 3, ( $\alpha, \beta$ )-privacy holds iff $\theta_{1}$ (struct $) \sim \ldots \sim \theta_{n}$ (struct). The models of $\alpha$ can be characterized by permutations: $\theta_{1}, \theta_{2}$ are models of $\alpha$ iff there is a permutation $\pi$ of $\{1, \ldots, N\}$ such that $\theta_{1}\left(v_{i}\right)=\theta_{2}\left(v_{\pi(i)}\right)$ for all $i \in\{1, \ldots, N\}$. The theorem now follows from the fact that all permutations can be obtained from each other by a sequence of swaps, and $\sim$ is an equivalence relation.
Example 11. Let us consider the setting of Definition 12 and Example 10 and let, for concreteness, the number of cast votes be $N=3$ and the number of votes for 1 be $R=2$. Then the privacy goal for binary votes is defined by

$$
\alpha \equiv \alpha_{a x} \wedge v_{1} \in\{0,1\} \wedge v_{2} \in\{0,1\} \wedge v_{3} \in\{0,1\} \wedge v_{1}+v_{2}+v_{3}=2
$$

where $v_{1}, v_{2}, v_{3}$ are variables and $\alpha_{a x}$ is the axiom characterizing addition. The models of $\alpha$ are $\Theta=\left\{\theta_{1}, \theta_{2}, \theta_{3}\right\}$ with $\theta_{1}=\left\{v_{1} \mapsto 1, v_{2} \mapsto 1, v_{3} \mapsto 0\right\}, \theta_{2}=$ $\left\{v_{1} \mapsto 1, v_{2} \mapsto 0, v_{3} \mapsto 1\right\}$ and $\theta_{3}=\left\{v_{1} \mapsto 0, v_{2} \mapsto 1, v_{3} \mapsto 1\right\}$.

Let, without loss of generality, the true result of the vote be $\theta=\theta_{1}=\left\{v_{1} \mapsto\right.$ $\left.1, v_{2} \mapsto 1, v_{3} \mapsto 0\right\}$, so that $\beta=\operatorname{MsgAna}(\alpha$, struct, $\theta)$ with

$$
\begin{aligned}
\text { struct }=\left\{\mid \mathrm{m}_{1}\right. & \mapsto \operatorname{crypt}\left(\operatorname{pub}(s), r_{1}, \operatorname{sign}\left(\operatorname{priv}(1), \operatorname{ballot}\left(v_{1}\right)\right)\right), \\
& \mathrm{m}_{2} \mapsto \operatorname{crypt}\left(\operatorname{pub}(s), r_{2}, \operatorname{sign}\left(\operatorname{priv}(2), \operatorname{ballot}\left(v_{2}\right)\right)\right), \\
& \mathrm{m}_{3} \\
& \mapsto \operatorname{crypt}\left(\operatorname{pub}(s), r_{3}, \operatorname{sign}\left(\operatorname{priv}(3), \operatorname{ballot}\left(v_{3}\right)\right)\right), \\
\mathrm{m}_{4} & \left.\mapsto 0, \mathrm{~m}_{5} \mapsto 1, \mathrm{~m}_{6} \mapsto \operatorname{pub}(s) \mid\right\}
\end{aligned}
$$

As we remarked above, $(\alpha, \beta)$-privacy holds since the intruder cannot analyze any messages and cannot compare them, thus every model of $\alpha$ can be extended to a model of $\beta$. We can also use Theorem 4 (and Theorem 3) to argue that ( $\alpha, \beta$ )-privacy holds iff $\theta_{1}($ struct $) \sim \theta_{2}($ struct $) \sim \theta_{3}($ struct $)$. It is easy to see that the models of $\alpha$ are characterized by permutations of $\{1,2,3\}$, and that all these permutations can be obtained from each other by a sequence of swaps. Hence, we can conclude by the fact that $\sim$ is an equivalence relation.

Theorem 4 shows that ( $\alpha, \beta$ )-privacy is actually a declarative logical characterization of the privacy goals, in contrast to the more technical vote-swapping formulation. Note also that privacy for some more advanced voting systems cannot be characterized by vote swapping, but such systems have a declarative specification using ( $\alpha, \beta$ )-privacy. For example, consider the following generalization of voting privacy:
Definition 13 (General Vote Privacy). Consider a voting system with $K$ candidates $\left\{c_{1}, \ldots, c_{K}\right\}$ and $N$ voters, and such that every voter has $L$ votes (one can give multiple votes to candidates). We use a binary interpreted function symbol $v[\cdot][\cdot]$ as follows: $v[i]\left[c_{j}\right]$ is the number of votes that voter $i$ has given to
candidate $c_{j}$. Then, publishing the total number of votes $R_{1}, \ldots, R_{K}$ for each candidate gives rise to the following privacy goal:

$$
\begin{aligned}
\alpha \equiv \alpha_{a x^{\prime}} & \wedge v[1]\left[c_{1}\right]+\ldots+v[1]\left[c_{K}\right]=L \wedge \ldots \wedge v[N]\left[c_{1}\right]+\ldots+v[N]\left[c_{K}\right]=L \\
& \wedge v[1]\left[c_{1}\right]+\ldots+v[N]\left[c_{1}\right]=R_{1} \wedge \ldots \wedge v[1]\left[c_{K}\right]+\ldots+v[N]\left[c_{K}\right]=R_{K}
\end{aligned}
$$

This shows that $(\alpha, \beta)$-privacy indeed provides us with a new and declarative way to formulate privacy goals also in the context of e-voting, and sets the basis for the modeling and analysis of full voting protocols, which will also require us to consider $(\alpha, \beta)$-privacy in the context of transition systems.

## 6 ( $\alpha, \beta$ )-privacy in Transition Systems

We now briefly show how we can extend ( $\alpha, \beta$ )-privacy to transition systems. Given that transition systems are not the main focus of this paper (and will be investigated in detail in future work) we here only discuss the key idea and a detailed example of two security protocols for private authentication.

The key idea is that we can define a state as a triple $(\alpha, \beta, \gamma)$ of formulae, where $\gamma$ represents the "truth". Then we can specify transition systems for this kind of states, and privacy is the question whether ( $\alpha, \beta$ )-privacy holds in every reachable state $(\alpha, \beta, \gamma)$. Formally, with $\Sigma, \Sigma_{0} \subseteq \Sigma, \mathcal{V}$ and $\approx$ as before:

Definition 14 (Transition systems). A state is a triple ( $\alpha, \beta, \gamma$ ), where $\alpha$ and $\beta$ are as before and $\gamma \in \mathcal{L}_{\Sigma_{0}}(\mathcal{V})$ is such that $\gamma \models \alpha$ and $\gamma$ is true in exactly one model of $\alpha$ (with respect to $\Sigma_{0}$ and the free variables of $\alpha$ ). We also call $\gamma$ the truth and may also apply it to $\Sigma_{0}$-terms like a substitution.

Let $\mathcal{S}$ denote the set of all states. A transition system is a pair $(I, R)$ where $I \in \mathcal{S}$ and $R \subseteq \mathcal{S} \times \mathcal{S}$. As is standard, the set of reachable states is the smallest set that contains $I$ and that is closed under $R$, i.e.: if $S$ is reachable and $\left(S, S^{\prime}\right) \in R$, then also $S^{\prime}$ is reachable.

We say that a transition system satisfies privacy iff ( $\alpha, \beta$ )-privacy holds in every reachable state $(\alpha, \beta, \gamma)$.
Example 12. As an example of privacy as reachability, consider a simple transition system with an initial state that has no information, and four successor states $S_{i, j}$ with $i, j \in\{0,1\}$ depending on two independent choices $i$ and $j$ of the user. In all four states, we have $\alpha \equiv x \in\{0,1\}$. Let now $\beta_{i, j} \equiv \operatorname{MsgAna}(\alpha$, struct, $\theta)$ where struct $=\left\{\mathrm{m}_{1} \mapsto \operatorname{scrypt}\left(\mathrm{k}_{j}, x\right), \mathrm{m}_{2} \mapsto \mathrm{k}_{1}\right\}$, $\mathrm{k}_{j}$ are new constants, and $\theta=\{x \mapsto i\}$.

In the states with $j=0$, the intruder cannot deduce anything interesting as he does not have the key needed for decryption, but in the states with $j=1$, we have $\beta_{i, 1} \models x=i$. So, there are reachable states in which the intruder can find out more than he is supposed to.

For an entire protocol one would not say that it satisfies its privacy goals if in every state ( $\alpha, \beta$ )-privacy holds; one may then, by slight abuse of notation, say $(\alpha, \beta)$-privacy holds for the protocol (referring to $(\alpha, \beta)$-privacy as a concept, not with respect to a concrete pair of formulas $\alpha$ and $\beta$ ). That both $\alpha$ and $\beta$ may grow over time should not be surprising, since the information that the intruder gathers may increase over time. For instance, at the beginning of a voting process, the result is not published yet (even if all voters have at this point already made up their mind how they want to vote). Moreover, a violation in an intermediate state of a system does not necessarily entail a violation in the end (votes are actually released at the end but must be protected while the election is being carried out).

Let us now consider two more sophisticated examples.

### 6.1 Analysis of Two Example Protocols

To illustrate further the expressiveness and strength of $(\alpha, \beta)$-privacy in transition systems, we consider two protocols proposed by Abadi and Fournet in [5]. The protocols are supposed to establish shared secrets between two agents $A$ and $B$. As they write: "The first protocol uses digital signatures and requires that principals have loosely synchronized clocks. The second protocol uses only encryption and avoids the synchronization requirement, at the cost of an extra message. The second protocol draws attention to difficulties in achieving privacy against an active adversary." More information on the protocols can be found in [5]; here, we will focus only on what is needed to show $(\alpha, \beta)$-privacy at work.

We adopt, whenever possible, Abadi and Fournet's notions and notations but introduce our own when needed. Consider a set of agents Agent and sets of agents $S_{A}$ for every $A \in$ Agent, representing the set of agents that $A$ is willing to talk to. In contrast to Abadi and Fournet, we write $\operatorname{pub}(A)$ and $\operatorname{priv}(A)$ for the public and private key of agent $A$, respectively.

### 6.1.1 The First Protocol (AF1)

The first protocol, which we call AF1, is as follows:
$A \rightarrow B:[\operatorname{hello}, \operatorname{crypt}(\operatorname{pub}(B),[\operatorname{hello}, \operatorname{pub}(A), \operatorname{sign}(\operatorname{priv}(A),[\operatorname{pub}(A), \operatorname{pub}(B), K, T])])]$
where hello is a tag, $K$ is a symmetric key freshly generated by $A, T$ is a timestamp (we assume that $B$ buffers all messages as long as their timestamp is considered recent, so replays can be detected) and $\left[t_{1}, \ldots, t_{n}\right]$ denotes the pairing pair $\left(t_{1}, \operatorname{pair}\left(t_{2}, \operatorname{pair}\left(\ldots, t_{n}\right)\right)\right)$.

Upon receipt of the message, the recipient $B$ tries to decrypt the second component using its private key $\operatorname{priv}(B)$. If the decryption yields a key $\operatorname{pub}(A)$ and a signed statement of the form $\operatorname{sign}(\operatorname{priv}(A),[\operatorname{pub}(A), \operatorname{pub}(B), K, T])])$, then $B$ extracts $\operatorname{pub}(A)$ and $K$, verifies the signature using $\operatorname{pub}(A)$, ensures that the message is not a replay using the timestamp $T$ and checks that $A \in S_{B}$. If

- the form does not check out (it is not encrypted with pub $(B)$, etc.),
- the message is a replay, or
- $A \notin S_{B}$ (i.e., $B$ is not willing to talk to $A$ ),
then $B$ should simply discard the message.
The encryptions are assumed to be which-key concealing, i.e., an intruder cannot tell which public key a message is encrypted with (unless he has the private key).


### 6.1.2 The Second Protocol (AF2)

The second protocol, which we call AF2, consists of two steps:

$$
\begin{array}{ll}
A \rightarrow B: & {\left[\text { hello, } \operatorname{crypt}\left(\operatorname{pub}(B),\left[\text { hello, } N_{A}, \operatorname{pub}(A)\right]\right)\right]} \\
B \rightarrow A: & \text { if valid then }\left[\operatorname{ack}, \operatorname{crypt}\left(\operatorname{pub}(A),\left[\operatorname{ack}, N_{A}, N_{B}, \operatorname{pub}(B)\right]\right)\right] \text { else }[\operatorname{ack}, R]
\end{array}
$$

where $N_{A}$ and $N_{B}$ are nonces and ack is an acknowledgment tag. We use an if-then-else to express that there is an error handling. $B$ checks that the message (that apparently comes from $A$ ) is not a replay and tries to decrypt the second component using its private key. If the decryption succeeds, then $B$ extracts the nonce $N_{A}$ and key $p u b(A)$, and checks that $A \in S_{B}$. If all these succeed, then the message is valid and $B$ generates a nonce $N_{B}$, and sends a reply to $A$. If $B$ receives an ill-formed message, then it will reply anyway, but with the second message in the "else" branch where $R$ is a random value. Note that the original paper has here encryption of a new nonce with another public key of $B$, but the point is simply that without knowing priv $(A)$, one should not be able to tell whether a given message was produced by the "then" or by the "else" branch. We thus assume here something even slightly stronger than which-keyconcealing: that also a random value is indistinguishable from the ciphertext, a property that at least holds in our algebraic model. ${ }^{9}$

Abadi and Fournet consider, among others, the following goal. If $A$ wishes to communicate with $B$, but not vice versa, then the intruder should not learn anything. Thus, a run between the two agents $A$ and $B$ should be indistinguishable from a run between two other agents $A^{\prime}$ and $B^{\prime}$ under some hypotheses. These hypotheses should include that $B$ is not the intruder and what can happen outside the protocol, i.e., what the agents can do besides running the protocol, which can result in leaks not caused by the protocol itself (see [5]).

### 6.1.3 Formalization of AF1 with ( $\alpha, \beta$ )-privacy

In order to formalize AF 1 with $(\alpha, \beta)$-privacy, let the payload alphabet $\Sigma_{0}$ consist of

[^9]- a set of agent names (finite or countably infinite),
- a binary relation $\operatorname{talk}(a, b)$ representing that $a$ is willing to talk to $b$ (in the notation of Abadi and Fournet: $b \in S_{a}$ ), and
- a unary predicate honest $(a)$ to identify a subset of agents $a$ that are honest (and thus follow the protocol entirely); all other (dishonest) agents are just marionettes of the intruder.

In the previous sections, we have used a substitution $\theta$ to characterize an arbitrary model of $\alpha$. Now we also have to interpret the predicates honest and talk, and therefore use a $\Sigma_{0}$-formula $\gamma$ instead of $\theta$. We will also use $\gamma$ sometimes as a substitution; by construction we will ensure that whenever $\alpha$ in a state contains a variable $x$, then $\gamma$ implies $x=a$ for some $a \in \Sigma_{0}$.

Definition 15. Let $\gamma_{0}$ be a closed $\Sigma_{0}$-formula with exactly one model (i.e., an arbitrary interpretation for the predicates honest and talk). The initial state $\left(\alpha_{0}, \beta_{0}, \gamma_{0}\right)$ of the transition system is as follows. Let $M_{0}=\{a \mid a \in$ $\left.\Sigma_{0}\right\} \cup\left\{\operatorname{pub}(a) \mid a \in \Sigma_{0}\right\} \cup\left\{\operatorname{priv}(d) \mid \gamma_{0} \not \models\right.$ honest $\left.(d)\right\}$ be the set of ground messages initially known by the intruder. Let struct ${ }_{0}=\left\{\mathrm{m}_{1} \mapsto t_{1}, \ldots \mathrm{~m}_{l} \mapsto t_{l}\right\}$ if $t_{1}, \ldots, t_{n}$ are the elements of $M_{0}$. Define

$$
\begin{aligned}
\alpha_{0} \equiv & \bigwedge\left\{\operatorname{talk}(a, b) \mid \gamma_{0} \models \operatorname{talk}(a, b) \wedge \neg \operatorname{honest}(a)\right\} \wedge \\
& \bigwedge\left\{\neg \operatorname{talk}(a, b) \mid \gamma_{0} \models \neg \operatorname{talk}(a, b) \wedge \neg \operatorname{honest}(a)\right\} \\
\beta \equiv & \operatorname{MsgAna}\left(\alpha_{0}, \operatorname{struct} t_{0}, \gamma_{0}\right)
\end{aligned}
$$

Note that $\alpha_{0}$ is ground.
Observe that the initial state preserves $(\alpha, \beta)$-privacy, since there are no variables, and struct $\sim \gamma_{0}$ (struct) in all models, thus it is consistent for all models of $\alpha$.

There is only one kind of transition:
Definition 16. The reachable states of AF1 are the least set of states that include the initial state and that are closed under the following transition rule. We use as an invariant that $\beta$ in every reachable state is a message-analysis problem. If $(\alpha, \beta, \gamma)$ is a reachable state with $\beta=\operatorname{MsgAna}(\alpha$, struct, $\gamma)$, then also the following state $\left(\alpha^{\prime}, \beta^{\prime}, \gamma^{\prime}\right)$ is reachable. Let $a$ and $b$ be any agents in $\Sigma_{0}$ such that $\gamma \vDash \operatorname{talk}(a, b) \wedge$ honest $(a)$. Let $x$ and $y$ be two fresh variables (that do not occur in $\alpha$ and $\beta$ ). Let $k$ and $t$ be arbitrary new uninterpreted constants of $\Sigma \backslash \Sigma_{0}$, and

$$
\begin{aligned}
\text { struct }^{\prime}= & \text { struct } \cup \\
& \left\{\mathrm{m}_{l+1} \mapsto[\operatorname{hello}, \operatorname{crypt}(\operatorname{pub}(y),[\operatorname{hello}, \operatorname{pub}(x), \operatorname{sign}(\operatorname{priv}(x),[\operatorname{pub}(x), \operatorname{pub}(y), k, t])])]\right\},
\end{aligned}
$$

where $l$ is the length of the frame struct. Define

$$
\begin{aligned}
\gamma^{\prime} & \equiv \gamma \wedge x=a \wedge y=b \\
\alpha^{\prime} & \equiv \alpha \wedge \operatorname{talk}(x, y) \wedge \begin{cases}\wedge\{y \neq c \mid \gamma \models \neg \operatorname{honest}(c)\} & \text { if } \gamma \models \text { honest }(b) \\
x=a \wedge y=b & \text { otherwise }\end{cases} \\
\beta^{\prime} \equiv & \operatorname{MsgAna}\left(\alpha^{\prime}, \text { struct }, \gamma^{\prime}\right)
\end{aligned}
$$

Note that we describe only transitions that are made by honest agents (since the other agents are just intruder marionettes). That is, the intruder learns that there is an agent $x$ who is willing to talk to $y$ and has started a session; additionally, if $y$, i.e. $b$, is dishonest, then the intruder knows who $x$ and $y$ are; if $y$ is honest, however, then the intruder only learns that $y$ is none of the dishonest agents. In order to generate an $(\alpha, \beta)$-privacy transition system we could, among other options, augment a process calculus notation in a suitable way. Then, to connect a process calculus notation with $(\alpha, \beta)$-privacy here (for this transition) we would need to explicitly denote in the process calculus which information is released at this point, namely that $\operatorname{talk}(x, y)$ is released to the public (i.e., to all) for arbitrary $x$ and $y$, and $x=a$ and $y=b$ is released to $y$ if $x$ or $y$ is dishonest. All the other information could be generated automatically in the process calculus notation.

Since $(\alpha, \beta)$-privacy is not based on distinguishability, but rather a reachability problem, the proof of privacy is in fact a pretty straightforward induction proof:

Lemma 4. Every reachable state of AF1 preserves ( $\alpha, \beta$ )-privacy.
Proof. First note that in no state the intruder learns the private key of an honest agent. The initial state preserves privacy as already noted. Let ( $\alpha, \beta, \gamma$ ) be any state in which privacy already holds and $\left(\alpha^{\prime}, \beta^{\prime}, \gamma^{\prime}\right)$ be any state that can be reached by one transition according to Definition 16. By the property of a reachable state, $\gamma$ describes a single model of $\alpha$. Moreover, $x$ and $y$ are exactly the new variables of $\alpha^{\prime}$ and $\operatorname{talk}(a, b)$ holds. Thus, $\gamma^{\prime}$ describes a single model of $\alpha^{\prime}$.

To see that $\beta^{\prime}$ is consistent for every model, we distinguish whether $b$ is honest or not. If $\gamma \models \operatorname{honest}(y)$, then the intruder does not know priv(b) (respectively priv $(y)$ ). Hence, the only check he can make (using the vcrypt function) is that the encrypted part cannot be decrypted with the private key of any dishonest agent, and thus that $y \neq c$ for any dishonest $c$. Since that is part of $\alpha^{\prime}$, this cannot produce an inconsistency.

If $\gamma \vDash \neg$ honest ( $y$ ), then $\alpha^{\prime}$ already implies $x=a$ and $y=b$ and therefore $\beta^{\prime} \models \operatorname{concr}[l+1]=\operatorname{struct}[l+1]$ where concr $=\gamma($ struct $)$. Thus, the addition to $\beta^{\prime}$ cannot produce an inconsistency either and $(\alpha, \beta)$-privacy is preserved by every reachable state of AF1.

### 6.1.4 Formalization of AF2 with ( $\alpha, \beta$ )-privacy

For AF2, we have the same setup and initial state; only the transitions are different. The first transition rule is as follows.

Definition 17. If $(\alpha, \beta, \gamma)$ is a reachable state with $\beta=\operatorname{MsgAna}(\alpha$, struct, $\gamma)$, then also the following state $\left(\alpha^{\prime}, \beta^{\prime}, \gamma^{\prime}\right)$ is reachable. Let $a$ and $b$ be any agents in $\Sigma_{0}$ such that $\gamma \models \operatorname{talk}(a, b) \wedge$ honest $(a)$. Let $x$ and $y$ be two fresh variables (that do not occur in $\alpha$ and $\beta$ ) and $n_{a}$ be a fresh constant. Let

$$
\text { struct }^{\prime}=\text { struct } \cup\left\{\mid \mathrm{m}_{l+1} \mapsto\left[\text { hello, } \operatorname{crypt}\left(\operatorname{pub}(y),\left[\text { hello, } n_{a}, \operatorname{pub}(x)\right]\right)\right]\right\},
$$

where $l$ is the length of the frame struct. Define

$$
\begin{array}{rlr}
\gamma^{\prime} & \equiv \gamma \wedge x=a \wedge y=b & \\
\alpha^{\prime} & \equiv \alpha \wedge \operatorname{talk}(x, y) \wedge \begin{cases}\wedge\{y \neq c \mid \gamma \models \neg \text { honest }(c)\} & \text { if } \gamma \models \text { honest }(b) \\
x=a \wedge y=b & \text { otherwise }\end{cases} \\
\beta^{\prime} & \equiv \operatorname{MsgAna}\left(\alpha^{\prime}, \text { struct }, \gamma^{\prime}\right) &
\end{array}
$$

In a process calculus notation, here we would as before need to make explicit how $\alpha^{\prime}$ extends $\alpha$ by the information that is released explicitly in this transition, namely $\operatorname{talk}(x, y)$ is released to the public (i.e., to all) for arbitrary $x$ and $y$, and $x=a$ and $y=b$ is released to $y$ if $y$ is dishonest, and otherwise, the intruder only learns that $y$ is none of the dishonest agents. All the other information could be generated automatically in the process calculus notation.

One may argue that $b$ cannot be sure at this point that it is really $a$ who contacted it here. However, we do not model that several dishonest agents cheat each other, but that they all work together as the intruder. Hence, our model shall simply not include a dishonest $a$ talking to a dishonest $b$ trying to cheat about its identity. Thus, it is safe to assume that whenever $b$ is dishonest and is apparently being contacted by $a$, it is indeed $a$ (being either honest or an accomplice).

We now have a second transition where some message is received by an agent $b$ and met with a reply (either the standard or the decoy message). We may assume that the message received by $b$ has a proper form, namely

$$
\left[\text { hello, } \operatorname{crypt}\left(\operatorname{pub}(B),\left[\text { hello, } N_{A}, \operatorname{pub}(A)\right]\right)\right]
$$

for some agents $A$ and $B$ and some term $N_{A}$. This message is either constructed by the intruder or a replay of a message the intruder has seen before. A priori, the intruder does not know whether the agent is actually $b$ or somebody else (he can only guess; we assume the agent names to be all public). Thus, $b$ will also answer with a decoy message if $B \neq b$ or if $\neg \operatorname{talk}(B, A)$.

Definition 18. If $(\alpha, \beta, \gamma)$ is a reachable state with $\beta=\operatorname{MsgAna}(\alpha$, struct, $\gamma)$, where $l$ is the length of the frame struct, then also the following state $\left(\alpha^{\prime}, \beta^{\prime}, \gamma\right)$ is reachable. Let rp be any recipe such that

$$
\beta \models \operatorname{gen}(r p) \wedge \operatorname{struct}[r p]=\left[\text { hello, } \operatorname{crypt}\left(\operatorname{pub}(B),\left[\text { hello }, N_{A}, \operatorname{pub}(A)\right]\right)\right]
$$

for some terms $A, B$ and $N_{A}$, where $A$ and $B$ are either constants of $\Sigma_{0}$ or free variables of $\alpha$. Let further $b \in \Sigma_{0}$ with $\gamma \models$ honest(b), and $r$ and $n_{b}$ be fresh
constants (that do not appear in $\beta$ ). Then:

$$
\begin{aligned}
\alpha^{\prime} \equiv & \alpha \wedge \begin{cases}B=b \wedge \operatorname{talk}(B, A) & \text { if } \gamma \models B=b \wedge \operatorname{talk}(B, A) \wedge \neg \text { honest }(A) \\
\text { true } & \text { otherwise }\end{cases} \\
\beta^{\prime} \equiv & \exists s . \operatorname{MsgAna(\alpha ,\operatorname {struct}\cup \{ |\mathrm {m}_{l+1}\mapsto s\} ,\gamma ^{\prime }\cup \{ s\mapsto t\} )} \\
& \wedge\left(B=b \wedge \operatorname{talk}(B, A) \Longleftrightarrow s=\left[\operatorname{ack}, \operatorname{crypt}\left(\operatorname{pub}(A),\left[\operatorname{ack}, N_{A}, n_{b}, \operatorname{pub}(B)\right]\right)\right]\right) \\
t= & \begin{cases}{\left[\operatorname{ack}, \operatorname{crypt}\left(\operatorname{pub}(\gamma(A)),\left[\operatorname{ack}, N_{A}, n_{b}, \operatorname{pub}(\gamma(B))\right]\right)\right]} & \text { if } \gamma \models B=b \wedge \operatorname{talk}(B, A) \\
{[\operatorname{ack}, r]} & \text { otherwise }\end{cases}
\end{aligned}
$$

First note the difference in how the "case split" is handled: while the concrete message $t$ is either the concrete answer message or the concrete decoy message (depending on the condition $B=b \wedge \operatorname{talk}(B, A)$ ), for the structural information $s$ we literally include the case split into the formula $\beta$, i.e., the intruder does not know the structure of the new message a priori, but he knows that the structure is either the regular message (if the condition $B=b \wedge \operatorname{talk}(B, A)$ holds true), or the decoy message, otherwise. ${ }^{10}$

The formula $\alpha^{\prime}$ permits the intruder to learn that $B=b$ and $\operatorname{talk}(B, A)$ if that is the case and $A$ is a dishonest agent. ${ }^{11}$ Otherwise, i.e., if $B \neq b$ or $\neg \operatorname{talk}(B, A)$, the intruder shall learn nothing. The answer from $b$ is thus either the normal message or the decoy message, depending on whether $B=$ $b \wedge \operatorname{talk}(B, A)$.

Observe that, if $A$ is dishonest, then the intruder has composed the input message to $B$ himself. Then the intruder knows $\operatorname{priv}(A)$ and can thus check whether the second part of the new message is (concretely) encrypted with $\operatorname{pub}(A)$. If that is the case, then it cannot be the decoy message, i.e., the intruder can derive $B=b \wedge \operatorname{talk}(B, A)$ because of $\phi_{\digamma_{1} \sim \digamma_{2}}$, but all this information is also part of $\alpha^{\prime}$.

If the decryption check fails, then it must be the decoy message and because of $\phi_{\digamma_{1} \sim \digamma_{2}}, B \neq b \vee \neg \operatorname{talk}(B, A)$. This formula does not follow from $\alpha^{\prime}$ in general (since we can take this transition in the initial state) and we thus have a violation of ( $\alpha, \beta$ )-privacy. Actually, we cannot guarantee that the $A$ does not learn anything here; it must be that one of two things is the case: $A$ has guessed wrongly who $B$ is, or $B$ does not want to talk to $A$. In fact, repeating this, $A$ can guess a number of possible agent names and thus either find out who $B$ is or find out (if an exhaustive search is possible) that $\neg \operatorname{talk}(B, A)$.

[^10]As far as we can see the only way to correct this is to release this information in $\alpha^{\prime}$ :

Definition 19. Consider the second transition with this alternative $\alpha^{\prime}$ :

$$
\alpha^{\prime} \equiv \alpha \wedge \begin{cases}B=b \wedge \operatorname{talk}(B, A) & \text { if } \gamma \models B=b \wedge \operatorname{talk}(B, A) \wedge \neg \operatorname{honest}(A) \\ B \neq b \vee \neg \operatorname{talk}(B, A) & \text { if } \gamma \models(B \neq b \vee \neg \operatorname{talk}(B, A)) \wedge \neg \operatorname{honest}(A) \\ \operatorname{true} & \text { otherwise }\end{cases}
$$

Compare this with a different but similar situation: if the intruder tries to use online guessing for a login. We cannot entirely prevent it (we can only limit the number of passwords he can try within a given time) and also cannot prevent that with every failed attempt the intruder learns that the password he tried was not correct. With $(\alpha, \beta)$-privacy we have a declarative way to express that: each failed guess leads to an augmentation of $\alpha$.

Thus, $(\alpha, \beta)$-privacy forces us to make explicit that we are leaking here a bit of information (which may be tolerable) and may lead to the awareness that we should protect the agent responding. In online guessing, it is common that after a failed attempt, one must wait a few seconds before a new attempt can be made. Similarly, here, $B$ should pause a few seconds after any message. It is necessary to do so also in the successful case, since when $A$ is honest it shall not be observable for the intruder whether a message was successful.

Lemma 5. Every reachable state of AF2 with the modification of Definition 19 satisfies ( $\alpha, \beta$ )-privacy.

Proof. Let $(\alpha, \beta, \gamma)$ a reachable state in which $(\alpha, \beta)$-privacy holds. Let $\left(\alpha^{\prime}, \beta^{\prime}, \gamma^{\prime}\right)$ be state reached with one transition of Definition 17. First, $\gamma^{\prime}$ describes one single model of $\alpha^{\prime}$. Like in the first protocol, either $b$ is honest or it is dishonest. If $b$ is honest, then the intruder does not know priv $(b)$ and cannot check the encrypted part of the message, and thus does not learn anything. If $b$ is dishonest, then $\beta^{\prime} \models \operatorname{concr}[l+1]=\operatorname{struct}[l+1]$ for the new position $l+1$ (where again concr $=\gamma($ struct $))$ and thus it cannot lead to inconsistencies.

Let $\left(\alpha^{\prime}, \beta^{\prime}, \gamma^{\prime}\right)$ be a state reached with one transition of Definition 18 with the fix of Definition 19. We note again that $\gamma^{\prime}$ describes one single model of $\alpha^{\prime}$. If $A$ is honest, then the intruder does not know priv $(A)$ and thus cannot check whether the second part of the message at position $l+1$ is encrypted with $\operatorname{pub}(A)$ or is a decoy. Thus, the augmentation of $\beta^{\prime}$ is consistent. Otherwise, regardless of whether $\gamma \models B=b \wedge \operatorname{talk}(B, A)$, we have $\beta^{\prime} \models \operatorname{struct}[l+1]=\operatorname{concr}[l+1]$, and thus $\beta^{\prime}$ cannot introduce any inconsistencies with any model of $\alpha^{\prime}$.

In future work, we will provide a more detailed account of $(\alpha, \beta)$-privacy in transition systems, but we believe that these detailed examples already provide for a convincing case.

## 7 Background Knowledge

One may wonder what happens if the intruder can make use of some background knowledge that is outside our formal model. Consider the following simplistic example. In a small rural village, everybody votes for the conservative party until one day a young couple moves to the village, and in the next election there are two left-wing votes. It does not take much imagination to figure out who was it (at least with high probability). Thus, with a bit of background information (and in fact common sense) one may be able to deduce information that was not deliberately released and in fact invade privacy. However note that this "attack" does not depend on the voting system: the best voting system cannot prevent this to happen since the system has been actually designed to release the total number of votes each candidate or party received. Of course, we cannot prevent an intruder from combining all the knowledge that is available to him, and thus, the voting system is not to blame as long as it does not release more information than specified in $\alpha$.

The subtle question is however: given that we have verified a system to have ( $\alpha, \beta$ )-privacy, but the intruder has some additional background knowledge $\alpha_{0}$ that the formal model does not take into account, what guarantees do we have in the system then? Obviously, the intruder can derive anything that is implied by $\alpha \wedge \alpha_{0}$ - the best system cannot prevent that. But could it be that the intruder can actually derive even more by some subtle combination of the information in $\beta$ and $\alpha_{0}$ ? We now show that this is not the case: our notion of $(\alpha, \beta)$-privacy is stable under an arbitrary consistent intruder background knowledge, i.e., the intruder cannot derive more than $\alpha \wedge \alpha_{0}$ :

Theorem 5. Consider a pair $(\alpha, \beta)$ according to Definition 7 and let the intruder's background knowledge be any formula $\alpha_{0} \in \mathcal{L}_{\Sigma_{0}}(f v(\alpha))$ such that $\beta \wedge \alpha_{0}$ is consistent. If $(\alpha, \beta)$-privacy holds, then also $\left(\alpha \wedge \alpha_{0}, \beta \wedge \alpha_{0}\right)$-privacy holds.

Proof. Suppose it were not the case, i.e., consider

- a pair $(\alpha, \beta)$ such that $(\alpha, \beta)$-privacy holds,
- a background knowledge $\alpha_{0} \in \mathcal{L}_{\Sigma_{0}}(f v(\alpha))$ such that $\beta \wedge \alpha_{0}$ is consistent, and
- an $\alpha^{\prime} \in \mathcal{L}_{\Sigma_{0}}(f v(\alpha))$ such that $\beta \wedge \alpha_{0}=\alpha^{\prime}$ but $\alpha \wedge \alpha_{0} \not \vDash \alpha^{\prime}$. (Thus, $\alpha^{\prime}$ is a witness that ( $\alpha \wedge \alpha_{0}, \beta \wedge \alpha_{0}$ )-privacy does not hold.)

By Definition 7, $\beta$ must have the form $\beta \equiv \alpha \wedge \beta_{0}$ for some $\beta_{0}$. We can thus rewrite $\beta \wedge \alpha_{0} \models \alpha^{\prime}$ as $\beta_{0} \models \neg \alpha \vee \neg \alpha_{0} \vee \alpha^{\prime}$. Now

- $\beta_{0} \models \neg \alpha$ is absurd, since then $\beta$ would be inconsistent;
- $\beta_{0} \models \neg \alpha_{0}$ is also absurd, since $\beta \wedge \alpha_{0}$ must be consistent; and
- $\beta_{0} \models \alpha^{\prime}$ would entail a violation of $(\alpha, \beta)$-privacy as $\alpha \wedge \alpha_{0} \not \vDash \alpha^{\prime}$ and thus $\alpha \not \vDash \alpha^{\prime}$.

One may remark here that this also indicates why we get a reasonable model of privacy in quantitative systems without actually considering quantitative measures or probabilities: for a good voting system, for instance, it simply does not matter how likely the different outcomes are, the cryptography should treat them all the same, only the background knowledge may be biased, but that does not really matter for the system and its privacy properties then.

## 8 Concluding Remarks

We have introduced $(\alpha, \beta)$-privacy as, we believe, a simple and declarative way to specify privacy goals and reason about them: the intruder should not be able to derive any "non-technical" statement from the technical information $\beta$ that he cannot derive from the intentionally released information $\alpha$ already. We have given a variety of concrete examples that describe how ( $\alpha, \beta$ )-privacy can be used in practice.

Above we have already compared extensively with static equivalence: we have described the simplicity of specifying properties via the declarative approach of $(\alpha, \beta)$-privacy with respect to the more tricky specifications in the static equivalence approach, and we have investigated formally the close relationship of $(\alpha, \beta)$-privacy to static equivalence, proving in particular the equivalence of message-analysis problems with a corresponding finite set of static equivalence problems. This result entails that we can use existing methods for deciding static equivalence for a given algebraic theory also for deciding the message-analysis fragment of $(\alpha, \beta)$-privacy for that theory.
( $\alpha, \beta$ )-privacy is course also related to observational equivalence (see, e.g., [8] as well as works on trace equivalence [12, 13, 16, 20]). In this approach, one typically considers labeled bi-similarity of two processes, checking that every transition of one process can be simulated by the other so that they are still bi-similar and so that their intruder knowledges are statically equivalent. This is difficult to automate but there are tricks that can be employed, such as turning it into a reachability problem by restricting the two processes to be the left and right variant of a bi-process. In contrast, thanks to the expressiveness of $(\alpha, \beta)$-privacy, we have a way to formulate privacy as a reachability problem while not being limited by bi-processes and the like. We believe that this gives opportunities for novel techniques for the automated verification of privacy goals.
( $\alpha, \beta$ )-privacy bears some similarities also with the non-interference approach and related works in information-flow and language-based security (see, e.g., $[24,35,32,33])$. Non-interference distinguishes (at least) two levels of information, usually low-level and high-variables. These are, however, fundamentally different from our payload $\alpha$ and technical information $\beta$ since they are formulae that express relations between values (rather than directly being public or private values). We actually do not mind that the intruder gets hold of (some) technical information as long as he cannot use it to obtain anything interesting besides the payload.

Privacy has also been studied in the area of statistical databases, building on database abstractions, in which records may contain identifiers, quasi-identifiers and sensitive attributes. Approaches in this field (such as $k$-anonymity, $\ell$ diversity, $t$-closeness and differential privacy) aim at quantifying privacy in order to capture privacy loss and thus analyze the minimal information disclosure inherent in a system. $k$-anonymity $[36,38]$ seeks to protect against identity disclosure by ensuring that a record is indistinguishable from $k-1$ other records on quasi-identifiers, usually replaying quasi-identifiers with equivalence classes of appropriate size. Hence, an intruder must be unable to reduce the anonymity set below a threshold of $k$ users. It is known that $k$-anonymity does not protect against attribute disclosure, which led to the development of $\ell$-diversity [31], i.e., the requirement that each equivalence class contains at least $\ell$ representations of a sensitive attribute. The work on $t$-closeness [29] observed that there are still attribute disclosures possible in $\ell$-diverse datasets, in particular, when the distribution of a sensitive attribute in an equivalence class is different from its distribution in the whole database, where this new notion stipulates that the distance between equivalence class and table distribution is at most $t$. Let us differentiate which statements over these three notions can be modeled in ( $\alpha, \beta$ )-privacy.

For $k$-anonymity, we observe that the property that $\alpha$ has at least $k$ models, and that the intruder cannot deduce an $\alpha^{\prime}$ with less choices, is in principle encodable in $(\alpha, \beta)$-privacy, which means that $(\alpha, \beta)$-privacy can express identity disclosure, although a full-fledged encoding of $k$-anonymity in $(\alpha, \beta)$-privacy will be subject of future work.

Our argument can be applied recursively to equivalence classes: for each equivalence class for any sensitive attribute, there are $\ell$ models, and the intruder cannot deduce an $\alpha^{\prime}$ with less choices for any equivalence class. Hence, the argument suggests also an ( $\alpha, \beta$ )-privacy encoding for the main definition of distinct $\ell$-diversity. However, $(\alpha, \beta)$-privacy does not have a notion of entropy and thus cannot encode entropy $\ell$-diversity.

Similarly, $(\alpha, \beta)$-privacy cannot encode directly $t$-closeness, which relies on distance of probability distributions (using variational distance or the KuhlbackLeibler distance on entropies as measure).

Finally, let us consider differential privacy [22], which asks whether an intruder can detect significant changes in a probability distribution on statistical data released by a curator on data sets differing in one element. As differential privacy is a property established on the information release function of the curator, a relation to our notion is not straightforward.

We have already mentioned above and in the previous sections a few directions for future work. In addition to these, we have already started to consider further examples than those discussed here, in particular examples that fall outside the message-analysis problem. To that end, we will need to generalize the definition of combinatoric $\alpha$ and to generalize our decidability results to larger fragments of $(\alpha, \beta)$-privacy. In fact, many interesting issues in e-voting fall outside the message-analysis fragment (and of the static equivalence approach).

We also plan to extend our formalization to a full-fledged specification of
( $\alpha, \beta$ )-privacy in transition systems. It will also be interesting to investigate how $(\alpha, \beta)$-privacy, which is a purely qualitative and possibilistic approach, can be extended to consider quantitative aspects of privacy such as: probabilities, time and cost of the private information.

Acknowledgement We thank Thomas Groß for our joint preliminary work and for many interesting discussions. This work was supported by the SapereAude project "Composec: Secure Composition of Distributed Systems", grant 4184-00334B of the Danish Council for Independent Research; the EU FP7 project no. 318424, "FutureID: Shaping the Future of Electronic Identity" (futureid.eu); the EU FP7 project no. 257876, "SPaCIoS: Secure Provision and Consumption in the Internet of Services" (www.spacios.eu); the EU H2020 project no. 700321 "LIGHTest: Lightweight Infrastructure for Global Heterogeneous Trust management in support of an open Ecosystem of Trust schemes" (lightest.eu); and the Italian PRIN 2010-11 project "Security Horizons".

## References

[1] M. Abadi. Private Authentication. In Privacy Enhancing Technologies, Second International Workshop, PET 2002, San Francisco, CA, USA, April 14-15, 2002, Revised Papers, LNCS 2482, pages 27-40. Springer, 2003.
[2] M. Abadi, M. Baudet, and B. Warinschi. Guessing Attacks and the Computational Soundness of Static Equivalence. Journal of Computer Security, 18:909-968, 2010.
[3] M. Abadi and V. Cortier. Deciding knowledge in security protocols under (many more) equational theories. In Proceedings of the Computer Security Foundations Workshop (CSFW), pages 62-76. IEEE CS Pr., 2005.
[4] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In Proceedings of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '01, pages 104-115. ACM Pr., 2001.
[5] M. Abadi and C. Fournet. Private authentication. Theoretical Computer Science, 322:427-476, 2004.
[6] O. Almousa, S. Mödersheim, and L. Viganò. Alice and Bob: Reconciling Formal Models and Implementation. In Programming Languages with Applications to Biology and Security - Essays Dedicated to Pierpaolo Degano on the Occasion of His 65th Birthday, LNCS 9465, pages 66-85. Springer, 2015.
[7] M. Arapinis, V. Cortier, and S. Kremer. When Are Three Voters Enough for Privacy Properties? In Computer Security-ESORICS 2016-21st European Symposium on Research in Computer Security, Heraklion, Greece,

September 26-30, 2016, Proceedings, Part II, LNCS 9879, pages 241-260. Springer, 2016.
[8] M. Arapinis, J. Liu, E. Ritter, and M. Ryan. Stateful applied pi calculus: Observational equivalence and labelled bisimilarity. J. Log. Algebr. Meth. Program., 89:95-149, 2017.
[9] B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In Proceedings of the Computer Security Foundations Workshop (CSFW), pages 82-96. IEEE CS Pr., 2001.
[10] B. Blanchet, M. Abadi, and C. Fournet. Automated verification of selected equivalences for security protocols. Journal of Logic and Algebraic Programming, 75(1):3-51, 2008.
[11] J. R. Büchi and L. H. Landweber. Definability in the Monadic SecondOrder Theory of Successor. J. Symb. Log., 34(2):166-170, 1969.
[12] V. Cheval, H. Comon-Lundh, and S. Delaune. A procedure for deciding symbolic equivalence between sets of constraint systems. Inf. Comput., 255:94-125, 2017.
[13] R. Chrétien, V. Cortier, and S. Delaune. Decidability of Trace Equivalence for Protocols with Nonces. In Proceedings of the IEEE 28th Computer Security Foundations Symposium, CSF 2015, Verona, Italy, 13-17 July, 2015, pages 170-184. IEEE CS Pr., 2015.
[14] V. Cortier. Formal verification of e-voting: solutions and challenges. SIGLOG News, 2(1):25-34, 2015.
[15] V. Cortier. Electronic Voting: How Logic Can Help. In Implementation and Application of Automata - 22nd International Conference, CIAA 2017, Marne-la-Vallée, France, June 27-30, 2017, Proceedings, LNCS 10329, pages xi-xii. Springer, 2017.
[16] V. Cortier, A. Dallon, and S. Delaune. SAT-Equiv: An Efficient Tool for Equivalence Properties. In 30th IEEE Computer Security Foundations Symposium, CSF 2017, Santa Barbara, CA, USA, August 21-25, 2017, pages 481-494. IEEE CS Pr., 2017.
[17] V. Cortier, N. Grimm, J. Lallemand, and M. Maffei. A Type System for Privacy Properties. In the 2017 ACM Pr. SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017, pages 409-423. ACM Pr., 2017.
[18] V. Cortier, M. Rusinowitch, and E. Zalinescu. Relating two standard notions of secrecy. Logical Methods in Computer Science, 3(3), 2007.
[19] V. Cortier and C. Wiedling. A formal analysis of the Norwegian E-voting protocol. Journal of Computer Security, 25(1):21-57, 2017.
[20] S. Delaune and L. Hirschi. A survey of symbolic methods for establishing equivalence-based properties in cryptographic protocols. J. Log. Algebr. Meth. Program., 87:127-144, 2017.
[21] S. Delaune, M. D. Ryan, and B. Smyth. Automatic verification of privacy properties in the applied pi-calculus. In Trust Management II: Proceedings of IFIPTM 2008: Joint iTrust and PST Conferences on Privacy, Trust Management and Security, June 18-20, 2008, Trondheim, Norway, IFIP 263, pages 263-278. Springer, 2008.
[22] C. Dwork. Differential Privacy: A Survey of Results. In Theory and Applications of Models of Computation, 5th International Conference, TAMC 2008, Xi'an, China, April 25-29, 2008 Proceedings, LNCS 4978, pages 119. Springer, 2008.
[23] H.-D. Ebbinghaus, J. Flum, and W. Thomas. Mathematical logic. Springer, 1994.
[24] J. A. Goguen and J. Meseguer. Security Policies and Security Models. In IEEE Symposium on Security and Privacy, Oakland, CA, USA, April 26-28, 1982, pages 11-20. IEEE CS Pr., 1982.
[25] J. Goubault-Larrecq. Finite models for formal security proofs. J. Comput. Secur., 18(6):1247-1299, 2010.
[26] P. Hankes Drielsma, S. Mödersheim, and L. Viganò. A Formalization of Off-Line Guessing for Security Protocol Analysis. In Logic for Programming, Artificial Intelligence, and Reasoning, 11th International Conference, LPAR 2004, Montevideo, Uruguay, March 14-18, 2005, Proceedings, LNCS 3452, pages 363-379. Springer, 2005.
[27] T. Hinrichs and M. Genesereth. Herbrand logic. Technical Report LG-2006-02, Stanford University, CA, USA, 2006. http://logic.stanford.edu/reports/LG-2006-02.pdf.
[28] IBM Research. Specification of the identity mixer cryptographic library. version 2.3.4. Technical report, IBM Research - Zurich, 2012.
[29] N. Li, T. Li, and S. Venkatasubramanian. t-Closeness: Privacy Beyond kAnonymity and l-Diversity. In Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007, pages 106-115. IEEE CS Pr., 2007.
[30] G. Lowe. Analysing protocols subject to guessing attacks. J. Comput. Secur., 12(1):83-97, 2004.
[31] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. ldiversity: Privacy beyond k-anonymity. ACM Transactions on Knowledge Discovery from Data (TKDD), 1, 2007.
[32] H. Mantel. Information flow and noninterference. In Encyclopedia of Cryptography and Security, pages 605-607. 2011.
[33] H. Mantel, D. Sands, and H. Sudbrock. Assumptions and Guarantees for Compositional Noninterference. In 24th IEEE Computer Security Foundations Symposium, CSF 2011, Cernay-la-Ville, France, 27-29 June, 2011, pages 218-232. IEEE CS Pr., 2011.
[34] S. Mödersheim, T. Groß, and L. Viganò. Defining Privacy Is Supposed to Be Easy. In Logic for Programming, Artificial Intelligence, and Reasoning - 19th International Conference, LPAR-19, Stellenbosch, South Africa, December 14-19, 2013. Proceedings, LNCS 8312, pages 619-635. Springer, 2013.
[35] P. Y. Ryan and S. Schneider. Process algebra and non-interference. In Proceedings of the Computer Security Foundations Workshop (CSFW). IEEE CS Pr., 1999.
[36] P. Samarati. Protecting Respondents' Identities in Microdata Release. IEEE Transactions on Knowledge and Data Engineering (TKDE), 13(6):1010-1027, 2001.
[37] P. Selinger. Models for an Adversary-Centric Protocol Logic. ENTCS, 55(1):69-84, 2003.
[38] L. Sweeney. k-Anonymity: A Model for Protecting Privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 10(5):557-570, 2002.
[39] C. Weidenbach. Towards an Automatic Analysis of Security Protocols in First-Order Logic. In Automated Deduction, CADE-16, 16th International Conference on Automated Deduction, Trento, Italy, July 7-10, 1999, Proceedings, LNCS 1632, pages 314-328. Springer, 1999.
[40] G. Zorn. RFC 2759: Microsoft PPP CHAP Extensions, Version 2, jan. 2000, 2000. Status: Informational.


[^0]:    General rights
    Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

    - Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
    - You may not further distribute the material or use it for any profit-making activity or commercial gain
    - You may freely distribute the URL identifying the publication in the public portal

    If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

[^1]:    ${ }^{1}$ There are also approaches, such as $k$-anonymity, $\ell$-diversity, $t$-closeness and differential privacy, that, instead of focusing on distinguishability, aim at quantifying privacy in order to capture privacy loss and thus analyse the minimal information disclosure inherent in a system. We will discuss these approaches, and their relationship to $(\alpha, \beta)$-privacy in more detail at the end of the paper (Section 8).

[^2]:    ${ }^{2}$ Note that, as will become clearer below, when we write "less technical" here we do not mean "less formal"; rather, we mean that our definition is declarative and not at the technical level like in the case of the different indistinguishability questions where one needs to consider explicitly the cryptographic messages that are being exchanged.

[^3]:    ${ }^{3}$ Note that it is common to define the Herbrand Universe as the free term algebra but for our purposes it is crucial to include also algebraic properties of the operators, as illustrated in Example 1.

[^4]:    ${ }^{4}$ It is, of course, common to use some quantifier-free representations, e.g., a set of clauses where all variables are implicitly universally quantified. However, in a logic with (arbitrarily scoped) quantifiers this is indeed non-standard. Consider, as an example, the absurd statement $x=c \models x \neq c$. In our definition of Herbrand Logic, this is false. If we consider all variables as explicitly quantified, i.e., $\forall x . x=c \models \forall x . x \neq c$, it is trivially true (since the formula on the left is unsatisfiable).

[^5]:    ${ }^{5}$ In this theory, every agent can have only one key pair for simplicity; to allow for arbitrary key infrastructures, one can rather model both pub and priv as public functions that map secret seeds to public and private keys, respectively.

[^6]:    ${ }^{6}$ The original definition of frames uses actually variables instead of constants for the labels, so that a frame is a substitution. In a logical context this is however quite inconvenient, since substitutions are meta level instead of object level, and mixing them usually does not lead to great results. Using constants for memory locations instead yields a clean solution and we can easily model frames as (object-level) functions on ground terms.

[^7]:    ${ }^{7}$ Note also that one may, of course, consider a similar use of variables for non-payload secrets, like the value $n$. However, since we require that $\alpha$ and $\beta$ have the same set of free variables, one would then existentially quantify that value, e.g.,

    $$
    \begin{aligned}
    & \beta \equiv \exists y . \ldots \quad \begin{array}{ll}
    \text { concr }\left[\mathrm{m}_{4}\right] & =\mathrm{h}(\operatorname{pair}(\mathrm{n}, \mathrm{a})) \wedge \operatorname{struct}\left[\mathrm{m}_{4}\right] \\
    \text { concr }\left[\mathrm{m}_{5}\right] & =\mathrm{h}\left(\operatorname{pair}\left(y, x_{1}\right)\right) \wedge \\
    (\operatorname{pair}(\mathrm{n}, \mathrm{~b})) \wedge \operatorname{struct}\left[\mathrm{m}_{5}\right] & =\mathrm{h}\left(\operatorname{pair}\left(y, x_{2}\right)\right)
    \end{array} \\
    & \text { concr }\left[\mathrm{m}_{5}\right]=\mathrm{h}(\operatorname{pair}(\mathrm{n}, \mathrm{~b})) \wedge \operatorname{struct}\left[\mathrm{m}_{5}\right]=\mathrm{h}\left(\operatorname{pair}\left(y, x_{2}\right)\right)
    \end{aligned}
    $$

    Without the existential quantifier (if $y$ were left free), the intruder could derive, e.g., that $y \neq \mathrm{a}$ (by generating $h\left(\operatorname{pair}\left(m_{1}, m_{1}\right)\right.$ ) and comparing the result with $\left.m_{4}\right)$. The $\exists$ thus intuitively says that we are not interested in the concrete value of $y$; in fact, the goal is not the protection of the nonces in the hash-values, so if they are found out, then it is not in itself a violation of privacy (but may lead to one).

[^8]:    ${ }^{8}$ Note that, in general, even for problems that fall outside the message-analysis fragment of ( $\alpha, \beta$ )-privacy there could still be suitable encodings into static equivalence problems.

[^9]:    ${ }^{9}$ If one wants to model that ciphertexts are recognizable, one can simply introduce a new operator vciph with the algebraic property $\operatorname{vciph}(\operatorname{crypt}(k, m)) \approx T$. Similarly one can model that a cipher is which-key-revealing with an operator vkey and the property $\operatorname{vkey}(k, \operatorname{crypt}(k, m)) \approx \top$.

[^10]:    ${ }^{10}$ Note that we do not specify transitions in Herbrand logic (just the states of the transition system are characterized by formulas in Herbrand logic), and so here we specified the transitions "by hand". Here we have modeled the case that the intruder uses a message for which he knows it has the right structure. This is clearly the case when he constructs such a message himself or if he uses one that was produced by an honest agent for the first step (because there is no decoy there). Sending any other message here necessarily would lead to a decoy and that is pointless for the intruder, so we omitted that here.
    ${ }^{11}$ In a process calculus notation, here we would need to make explicit how $\alpha^{\prime}$ extends $\alpha$ by the information that is released explicitly in this transition, namely that the information $B=b$ and $\operatorname{talk}(B, A)$ is released to the dishonest agent $A$.

