Guessing Attacks and the Computational Soundness of Static Equivalence

Martín Abadi[1], Mathieu Baudet[2], and Bogdan Warinschi[3]

[1] University of California, Santa Cruz
[2] LSV, CNRS & INRIA Futurs projet SECSI & ENS Cachan, France
[3] Loria, INRIA, Nancy, France

Abstract. The indistinguishability of two pieces of data (or two lists of pieces of data) can be represented formally in terms of a relation called static equivalence. Static equivalence depends on an underlying equational theory. The choice of an inappropriate equational theory can lead to overly pessimistic or overly optimistic notions of indistinguishability, and in turn to security criteria that require protection against impossible attacks or—worse yet—that ignore feasible ones. In this paper, we define and justify an equational theory for standard, fundamental cryptographic operations. This equational theory yields a notion of static equivalence that implies computational indistinguishability. Static equivalence remains liberal enough for use in applications. In particular, we develop and analyze a principled formal account of guessing attacks in terms of static equivalence.

1 Introduction

In the study of security, it is frequent to reason about whether two pieces of data can be distinguished by an observer. For example, the pieces of data might be two encrypted messages, and the observer an attacker that attempts to learn something about the underlying cleartexts by analyzing the encrypted messages. The two encrypted messages are indistinguishable if, no matter how the attacker operates on them, it cannot discern any meaningful difference. The encrypted messages may however be different—for instance, they may look like different random numbers.

Formally, indistinguishability can be represented in terms of a relation called static equivalence [4]. Roughly, two terms (and, more generally, two lists of terms) are statically equivalent when they satisfy all the same equations. This relation is essentially a special case of the observational equivalence relation of process calculi. It is simpler than observational equivalence in that it does not allow for continued interaction between a system and an observer: the observer gets data once and then conducts experiments on its own. Nevertheless, observational equivalence can be reduced to a combination of static equivalence and usual bisimulation requirements [4, 5, 15].

Static equivalence depends on an underlying equational theory. The choice of an inappropriate equational theory can lead to overly pessimistic or optimistic notions of indistinguishability, and in turn to security criteria that require protection against impossible attacks or—worse yet—that ignore feasible ones.

In this paper, we define an equational theory for standard, fundamental cryptographic operations, and we justify and apply the resulting concept of static equivalence. These operations include various flavors of encryption and decryption. Static equivalence in this theory implies computational indistinguishability. In other words, if the formal notion of static equivalence indicates that two pieces of data are indistinguishable, then no computationally feasible experiment can tell those two pieces of data apart. (This property is a soundness theorem. Although it is less important, we have also explored completeness, but we omit its discussion here; see however [28, 9].) Our notion of computational feasibility is based on the sorts of assumptions typically employed in complexity-theoretic cryptography. It includes certain assumptions on the security properties of the cryptographic operations; those assumptions appear reasonable and fairly standard, but so do others, and picking satisfactory ones is somewhat delicate.

While static equivalence is conservative enough to exclude feasible attacks, it also remains liberal enough for use in applications. In particular, we develop a formal account of guessing attacks (e.g., [23, 24, 13, 32, 31]) in terms of static equivalence. Since guessing attacks constitute a significant threat against protocols that rely on passwords and other weak secrets, the recent literature contains several studies of guessing attacks, with both formal and computational approaches (e.g., [27, 19, 18, 17, 10, 12, 16, 22, 25, 21]). Formal approaches are attractive because of their relative simplicity, which often enables automation. On the other hand, formal approaches are rather varied and sometimes ad hoc. Fortunately, it has been suggested that a formulation of guessing attacks could be based on static equivalence [17, 20]. We believe that this idea has a number of virtues. It leads to a crisp definition, it is fairly independent of specific choices of cryptographic operations, and it extends nicely to general process calculi. To date, however, this idea has not been worked out fully, in the setting of an appropriate equational theory. We aim to address this gap.

A related, frequent shortcoming of formal analyses is the lack of computational justifications. This lack allows the possibility that a protocol is safe against attacks formally, but that a feasible attack exists nonetheless. An active line of recent work aims to address such shortcomings, by defining and proving computational soundness results for formal methods (e.g., [6, 8, 29, 26]). That line of research includes a computational study of static equivalence [11]; the theories considered there do not include the one that we define in this paper (in part because those theories do not model probabilistic encryption functions, nor encryption under weak keys) and have not provided a satisfactory account of guessing attacks, but they are an important piece of the context of this work. That line of research also includes a study of guessing attacks, with an ad hoc formal definition of those attacks [7]. In this paper we build on that previous work, and go beyond it.

The next section, Section 2, presents a formal model: it defines sorted terms, an equational theory for them, and the corresponding notion of static equivalence. Section 3 interprets the syntax of the formal model in a computational universe; it includes cryptographic assumptions. Section 4 establishes the computational soundness of static equivalence for the equational theory. Section 5 applies our results to the study of guessing attacks. Section 6 concludes. Because of space constraints, we omit cryptographic constructions, a decision procedure for static equivalence, proofs, and additional details; these are included in an extended version of this paper [1].

2 Abstract Model

In order to represent cryptographic messages in an abstract way, we use terms over a many-sorted signature, equipped with an equational theory.

2.1 Sorts and Terms

The set of *sorts (or types)* that we consider is defined by the following grammar:

$\tau ::=$

$\mid$	$SKey$	symmetric keys
$\mid$	$EKey$	(public) encryption keys
$\mid$	$DKey$	(private) decryption keys
$\mid$	$Data$	passwords and other data
$\mid$	$Coins$	coins for encryption
$\mid$	$Pair[\tau_1, \tau_2]$	pairs of messages
$\mid$	$SCipher[\tau]$	symmetric encryptions of messages of type τ
$\mid$	$ACipher[\tau]$	asymmetric encryptions of messages of type τ

The set of *(well-sorted) terms*, written S, T, U, V, ..., is built from an infinite number of variables x, y, ... and names a, b, n, r, k, sk, pk, ... for each sort, with the following function symbols:

$\mathsf{enc}_\tau : \tau \times Data \to \tau$	encryption under data
$\mathsf{dec}_\tau : \tau \times Data \to \tau$	decryption with data
$\mathsf{penc}_\tau : \tau \times EKey \times Coins \to ACipher[\tau]$	public-key encryption
$\mathsf{pdec}_\tau : ACipher[\tau] \times DKey \to \tau$	private-key decryption
$\mathsf{pub} : DKey \to EKey$	public-key extraction
$\mathsf{pdec\_success}_\tau : ACipher[\tau] \times DKey \to Data$	domain predicate for private-key decryption
$\mathsf{senc}_\tau : \tau \times SKey \times Coins \to SCipher[\tau]$	symmetric encryption
$\mathsf{sdec}_\tau : SCipher[\tau] \times SKey \to \tau$	symmetric decryption
$\mathsf{sdec\_success}_\tau : SCipher[\tau] \times SKey \to Data$	domain predicate for symmetric decryption
$\mathsf{pair}_{\tau_1,\tau_2} : \tau_1 \times \tau_2 \to Pair[\tau_1, \tau_2]$	pairing
$\mathsf{fst}_{\tau_1,\tau_2} : Pair[\tau_1, \tau_2] \to \tau_1$	first projection
$\mathsf{snd}_{\tau_1,\tau_2} : Pair[\tau_1, \tau_2] \to \tau_2$	second projection
$0, 1 : Data$	boolean constants
$\mathsf{w}, \mathsf{c}_0, \mathsf{c}_1 \ldots : Data$	additional data constants

Encryption and decryption symbols may not be available for all sorts τ. We let T_{penc} be the set of types τ for which the symbols penc_τ, pdec_τ, and $\mathsf{pdec\_success}_\tau$ are available, and define T_{senc} and T_{enc} analogously. We assume that pairs are not encrypted under data values, that is, $T_{\mathsf{enc}} \cap \{Pair[\tau_1, \tau_2]\}_{\tau_1,\tau_2} = \emptyset$; pairs may however be encrypted with enc component by component.

Our function symbols represent encryption and decryption functions and auxiliary operations. The first two functions (enc_τ and dec_τ) are to be used with data values as keys; the data values may be the constant symbols of the grammar, which may represent the passwords in a dictionary. (In contrast, fresh names may represent strong keys; the scoping rules justify the respective uses of constant symbols and names.) The fact that enc_τ does not take a parameter of type *Coins* relates to the difficulties with probabilistic password-based encryption [7]. Moreover, the language provides no direct way for the attacker to check that a value results from applying enc_τ with a particular key. Such properties are essential for thwarting guessing attacks in practice (for example, in the EKE protocol [13]). The remaining functions are fairly standard; they include functions for public-key and symmetric encryption (penc_τ and senc_τ), which are probabilistic in the sense that they take a parameter of type *Coins*.

We often omit type annotations on function symbols. For instance, provided that S, T, and U have type *Data*, we may write $\mathsf{pair}(\mathsf{enc}(S,T),U)$ instead of $\mathsf{pair}_{Data,Data}(\mathsf{enc}_{Data}(S,T),U)$. In addition, we sometimes use the abbreviations $\{S\}_T$ for $\mathsf{enc}(S,T)$, $\{S\}^r_{\mathsf{pub}(sk)}$ for $\mathsf{penc}(S,\mathsf{pub}(sk),r)$, and $\{S\}^r_k$ for $\mathsf{senc}(S,k,r)$.

We write $\mathrm{var}(T)$ and $\mathrm{names}(T)$ for the sets of variables and names that occur in a term T. We extend the notation to tuples and sets of terms. A term T is *ground* or *closed* when $\mathrm{var}(T) = \emptyset$. We write $\sigma = \{x_1 \mapsto T_1, \ldots, x_n \mapsto T_n\}$ for a substitution, and let $\mathrm{dom}(\sigma) = \{x_1, \ldots, x_n\}$, $\mathrm{var}(\sigma) = \mathrm{var}(T_1, \ldots, T_n)$, and $\mathrm{names}(\sigma) = \mathrm{names}(T_1, \ldots, T_n)$. A substitution σ is *ground* or *closed* when $\mathrm{var}(\sigma) = \emptyset$. We consider only well-sorted substitutions (that is, for each i, $T_i = x_i\sigma$ has the same sort as x_i).

2.2 Equational Theory

We model the semantics of the cryptographic primitives by equipping terms with an equational theory, that is, a reflexive, symmetric, transitive relation, stable by (well-sorted) substitutions of terms for variables and (in this case) for names, and stable by application of contexts. Specifically, we consider the equational theory $=_E$ generated by the following equations:

$$\begin{array}{rr}
\mathsf{dec}_\tau(\mathsf{enc}_\tau(x,y),y) = x & \mathsf{enc}_\tau(\mathsf{dec}_\tau(x,y),y) = x \\
\mathsf{pdec}_\tau(\mathsf{penc}_\tau(x,\mathsf{pub}(y),z),y) = x & \mathsf{pdec_success}_\tau(\mathsf{penc}_\tau(x,\mathsf{pub}(y),z),y) = 1 \\
\mathsf{sdec}_\tau(\mathsf{senc}_\tau(x,y,z),y) = x & \mathsf{sdec_success}_\tau(\mathsf{senc}_\tau(x,y,z),y) = 1 \\
\mathsf{fst}_{\tau_1,\tau_2}(\mathsf{pair}_{\tau_1,\tau_2}(x,y)) = x & \mathsf{snd}_{\tau_1,\tau_2}(\mathsf{pair}_{\tau_1,\tau_2}(x,y)) = y \\
\mathsf{pair}_{\tau_1,\tau_2}(\mathsf{fst}_{\tau_1,\tau_2}(x),\mathsf{snd}_{\tau_1,\tau_2}(x)) = x &
\end{array}$$

where the symbols x, y, and z represent variables of the appropriate sorts. Most of the equations are fairly standard. The only surprise may be the inclusion of

$\mathsf{enc}_\tau(\mathsf{dec}_\tau(x,y),y) = x$, without which an attacker that sees x and guesses y might confirm whether x is a ciphertext encrypted under y by decrypting x with y, reencrypting with y, and comparing the result to x; the equation implies that the comparison always succeeds, whether the guess was correct or not. So, for instance, $\mathsf{enc}_\tau(n,\mathsf{c}_0)$ and $\mathsf{enc}_\tau(n,\mathsf{c}_1)$ are indistinguishable when n is a fresh name of sort τ. Such consequences of the equation are important for the security of protocols that rely on weak secrets. Moreover, the equation holds in many reasonable implementations, in particular those based on keyed permutations.

When oriented from left to right, the equations above form a convergent rewriting system that we call $\mathcal{R}$.

2.3 Frames and Static Equivalence

Frames represent sets of messages available to an observer (for example, because they were sent over a public network) [4]. More precisely, a *frame* is an expression $\varphi = \nu\tilde{n}.\{x_1 = T_1, \ldots, x_n = T_n\}$ where $\tilde{n}$ is a set of *restricted* names, and each T_i is a closed term of the same sort as x_i. For simplicity, we require (without loss of generality) that every name in use be restricted, that is, $\tilde{n} = \mathrm{names}(T_1, \ldots, T_n)$. A name k may still be disclosed explicitly, for instance by a dedicated mapping $x_i = k$. Therefore, we tend to omit the binders $\nu\tilde{n}$, and identify a frame φ with its *underlying substitution* $\{x_1 \mapsto T_1, \ldots, x_n \mapsto T_n\}$.

A closed term T is *deducible* from a frame φ if there exists a term M with $\mathrm{var}(M) \subseteq \mathrm{dom}(\varphi)$ and $\mathrm{names}(M) \cap \mathrm{names}(\varphi) = \emptyset$ such that $M\varphi =_E T$ [2,3].

Two frames φ_1 and φ_2 such that $\mathrm{dom}(\varphi_1) = \mathrm{dom}(\varphi_2)$ are *statically equivalent* (written $\varphi_1 \approx_E \varphi_2$) if, for every pair of terms (M,N) such that $\mathrm{var}(M,N) \subseteq \mathrm{dom}(\varphi_1)$ and $\mathrm{names}(M,N) \cap \mathrm{names}(\varphi_1,\varphi_2) = \emptyset$, it holds that $M\varphi_1 =_E N\varphi_2$ if and only if $M\varphi_2 =_E N\varphi_2$. Proving static equivalence may not be easy. Fortunately, efficient methods exist in many cases (e.g., [2,14]). In particular, static equivalence is decidable in polynomial time for unsorted convergent subterm theories [2]; we expect that this result carries over to sorted convergent subterm theories such as $=_E$. We have an alternative decision procedure for the static equivalences that are the subject of our main theorem (see Section 4).

We close this section with a few examples of equivalences and inequivalences under the theory E:

$$\{x = \{0\}_k^r\} \;\approx_E\; \{x = \{1\}_k^r\} \tag{1}$$
$$\{x = \{0\}_k^r,\;\; y = \{0\}_k^{r'}\} \;\approx_E\; \{x = \{1\}_k^r,\;\; y = \{0\}_{k'}^{r'}\} \tag{2}$$
$$\{x = \{n\}_{\mathsf{w}},\;\; y = \{m\}_{\mathsf{w}}\} \;\approx_E\; \{x = a_1,\;\; y = a_2\} \tag{3}$$
$$\{x = \{\{n\}_{\mathsf{w}}\}_{\mathsf{w}},\;\; y = \{m\}_{\mathsf{w}}\} \;\approx_E\; \{x = a_1,\;\; y = a_2\} \tag{4}$$
$$\{x = \{\{0\}^{r_1}_{\mathsf{pub}(sk)}\}_{\mathsf{w}},\;\; y = \{0\}^{r_2}_{\mathsf{pub}(sk)}\} \;\approx_E\; \{x = a_1,\;\; y = a_2\} \tag{5}$$
$$\{x = \{\{0\}^{r_1}_{\mathsf{pub}(sk)}\}_{\mathsf{w}},\;\; y = \{0\}^{r_1}_{\mathsf{pub}(sk)}\} \;\approx_E\; \{x = \{a_1\}_{\mathsf{w}},\;\; y = a_1\} \tag{6}$$
$$\{x = \{\{n\}_k^{r_1}\}_{\mathsf{w}},\;\; y = k\} \;\not\approx_E\; \{x = a_1,\;\; y = k\} \tag{7}$$

Examples (1) and (2) are simple examples about symmetric encryptions under strong keys, illustrating that those encryptions hide plaintexts and also equalities of plaintexts or keys across encryptions. Examples (3) and (4) illustrate that encryptions of fresh names under a constant w (intuitively, under a weak secret) can look like fresh names. The values of x and y are two such encryptions—and the former is in fact a double encryption in example (4)—with unrelated underlying names. Example (5) resembles example (4); it illustrates that an encryption of a public-key ciphertext $\{0\}^{r_1}_{\mathsf{pub}(sk)}$ under w can look like a fresh name. In examples (3)–(5), the plaintexts being encrypted are not otherwise available to the observer, though somewhat related plaintexts may be (as the values of the variable y). Example (6) treats a case in which the observer also obtains the plaintext being encrypted, through y; in that case, the observer can see a relation between the value of x and the value of y, namely that the former is an encryption of the latter under w. Example (7) indicates that the observer that is given k can distinguish $\{\{n\}^{r_1}_k\}_{\mathsf{w}}$ from a fresh name; intuitively, after decrypting with w, the adversary can tell if what it sees is a ciphertext under k or not, since the success of shared-key decryption is detectable.

3 Implementation

In this section we interpret the syntax of the formal model in a computational universe. We also discuss cryptographic assumptions on which the implementation relies.

3.1 Interpreting the Syntax

Next we detail the mapping from terms to distribution ensembles over bit-strings.

Encryption schemes. The mapping uses a public-key encryption scheme $\Pi^p = (\mathcal{K}^p, \mathcal{E}^p, \mathcal{D}^p)$ and a symmetric encryption scheme $\Pi^s = (\mathcal{K}^s, \mathcal{E}^s, \mathcal{D}^s)$. It also uses a symmetric, deterministic, type-preserving encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. (The definition of type preserving is given below.) In each of these triples, the first component is a key-generation function, the second an encryption function, and the third a decryption function. We write η for a security parameter. For each η, we write $k \xleftarrow{R} \mathcal{K}_\eta$ and $k \xleftarrow{R} \mathcal{K}^s_\eta$ for the process of generating an encryption key k for Π and Π^s, respectively, and similarly we write $(pk, sk) \xleftarrow{R} \mathcal{K}^p_\eta$ for the process of generating a pair (pk, sk) of encryption and decryption keys for Π^p. As usual, the encryption functions $\mathcal{E}^p$ and $\mathcal{E}^s$ are randomized; we write $\mathcal{E}^p(m, k, r)$ and $\mathcal{E}^s(m, k, r)$ for public-key and symmetric encryptions, respectively, of message m under encryption key k with random coins r. We write $c \xleftarrow{R} \mathcal{E}^p(m, k)$ and $c \xleftarrow{R} \mathcal{E}^s(m, k)$ for the corresponding encryption processes, using fresh random coins. We assume that the set of keys for Π is of the form $\{0,1\}^{\alpha_1(\eta)}$, and that the set of coins for Π^s and Π^p is $\{0,1\}^{\alpha_2(\eta)}$, where the functions $\alpha_1(\eta)$ and $\alpha_2(\eta)$ are polynomially bounded and at least linearly increasing.

We say that Π is *type-preserving* when, for every $\tau \in T_{\mathsf{enc}}$, encryption and decryption by Π map $[\![\tau]\!]_\eta$—the set of bit-strings that corresponding to the type τ—to itself.

Sorts, functions, and random drawings. For each value of the security parameter η, the concrete meaning of sorts and terms is characterized (much as in [11]) by:

- for each sort τ, a carrier set $[\![\tau]\!]_\eta$;
- for each function symbol $f : \tau_1 \times \ldots \times \tau_n \to \tau$, a function $[\![f]\!]_\eta : [\![\tau_1]\!]_\eta \times \ldots \times [\![\tau_n]\!]_\eta \to [\![\tau]\!]_\eta$;
- for each sort τ, a procedure written $e \xleftarrow{R} [\![\tau]\!]_\eta$ for drawing a random element e from $[\![\tau]\!]_\eta$, according to a distribution written $(\xleftarrow{R} [\![\tau]\!]_\eta)$.

We require that no element in $[\![\tau]\!]_\eta$ has probability 0 according to $(\xleftarrow{R} [\![\tau]\!]_\eta)$, that the probability of collision for $(\xleftarrow{R} [\![\tau]\!]_\eta)$ is negligible (that is, asymptotically smaller than any inverse polynomial), and that all the operations mentioned are computable in probabilistic polynomial time (PPTIME) in the complexity parameter. These conditions are ensured by the construction below and the properties of secure encryption schemes (defined in the next subsection).

The carrier set $[\![\tau]\!]_\eta$ of a type τ is defined inductively:

$$\begin{aligned}
[\![SKey]\!]_\eta &= \text{“}SKey\text{”} \parallel \{\text{symmetric keys for } \Pi^s(\eta)\}\\
[\![EKey]\!]_\eta &= \text{“}EKey\text{”} \parallel \{\text{public keys for } \Pi^p(\eta)\}\\
[\![DKey]\!]_\eta &= \text{“}DKey\text{”} \parallel \{\text{private keys for } \Pi^p(\eta)\}\\
[\![Data]\!]_\eta &= \text{“}Data\text{”} \parallel \{0,1\}^{\alpha_1(\eta)}\\
[\![Coins]\!]_\eta &= \text{“}Coins\text{”} \parallel \{0,1\}^{\alpha_2(\eta)}\\
[\![Pair[\tau_1,\tau_2]]\!]_\eta &= \text{“}Pair\text{”} \parallel [\![\tau_1]\!]_\eta \parallel [\![\tau_2]\!]_\eta\\
[\![SCipher[\tau]]\!]_\eta &= \text{“}SCipher\text{”} \parallel \tau \parallel \{\text{ciphertexts of } \Pi^s(\eta)\}\\
[\![ACipher[\tau]]\!]_\eta &= \text{“}ACipher\text{”} \parallel \tau \parallel \{\text{ciphertexts of } \Pi^p(\eta)\}
\end{aligned}$$

where $\parallel$ denotes the concatenation of bit-strings (applied by extension on sets of bit-strings), and we assume an encoding of identifiers for types τ into bit-strings.

The meaning of function symbols is as follows:

- Symbols $\mathsf{pair}_{\tau_1,\tau_2}$, $\mathsf{fst}_{\tau_1,\tau_2}$, and $\mathsf{snd}_{\tau_1,\tau_2}$ are implemented on bit-strings by tagged concatenation and projections, as one might expect.
- Constants w, c_0, c_1, ... are mapped to arbitrary PPTIME-computable sequences of bit-strings of length $\alpha_1(\eta)$, prefixed with the tag “*Data*”; 0 and 1 are mapped respectively to “*Data*” $\parallel 0^{\alpha_1(\eta)}$ and “*Data*” $\parallel 1^{\alpha_1(\eta)}$.
- For every $\tau \in T_{\mathsf{penc}}$, the implementations of penc_τ, pdec_τ, and $\mathsf{pdec\_success}_\tau$ are defined by:

$$[\![\mathsf{penc}_\tau]\!]_\eta(m, \text{“}EKey\text{”}||pk, \text{“}Coins\text{”}||r) = \text{“}ACipher\text{”}\|\tau\|\mathcal{E}^p_\tau(m, pk, r)$$

$$[\![\mathsf{pdec}_\tau]\!]_\eta(m, \text{“}DKey\text{”}||sk) = \begin{cases} \mathcal{D}^p(c, sk) & \text{if } m = \text{“}ACipher\text{”}\|\tau\|c \text{ and the} \\ & \text{decryption } \mathcal{D}^p(c, sk) \text{ succeeds} \\ \langle\text{any value}\rangle & \text{otherwise} \end{cases}$$

$$[\![\mathsf{pdec\_success}_\tau]\!]_\eta(m, \text{“}DKey\text{”}||sk) = \text{“}Data\text{”} \parallel \begin{cases} 1^{\alpha_1(\eta)} & \text{if } m = \text{“}ACipher\text{”}\|\tau\|c \text{ and the decryption } \mathcal{D}^p(c, sk) \text{ succeeds} \\ 0^{\alpha_1(\eta)} & \text{otherwise} \end{cases}$$

The implementations of senc_τ, sdec_τ, and $\mathsf{sdec\_success}_\tau$, for $\tau \in T_{\mathsf{senc}}$, are defined similarly.

– For every $\tau \in T_{\mathsf{enc}}$, the implementations of enc_τ and dec_τ are defined by:

$$\begin{aligned} [\![\mathsf{enc}_\tau]\!]_\eta(m, \text{“}\mathit{Data}\text{”} \| k) &= \mathcal{E}(m, k) \\ [\![\mathsf{dec}_\tau]\!]_\eta(c, \text{“}\mathit{Data}\text{”} \| k) &= \mathcal{D}(c, k) \end{aligned}$$

We assume that $\mathcal{E}(\cdot, k)$ and $\mathcal{D}(\cdot, k)$ are inverse bijections from $[\![\tau]\!]_\eta$ to itself. In particular, tags are left unchanged by these functions.

The drawing of random values of type τ ($e \xleftarrow{R} [\![\tau]\!]_\eta$) is defined by induction on τ (with, in addition, the appropriate tags in each case):

– When τ is one of SKey, EKey, and DKey, use the dedicated key generation algorithm, respectively $\mathcal{K}^s$, $\mathsf{fst}(\mathcal{K}^p)$, and $\mathsf{snd}(\mathcal{K}^p)$.
– When τ is Data or Coins, use the uniform distribution over $[\![\tau]\!]_\eta$.
– When $\tau = \mathit{Pair}[\tau_1, \tau_2]$, recursively draw random elements in $[\![\tau_1]\!]_\eta$ and $[\![\tau_2]\!]_\eta$, then tag and concatenate them.
– When τ is $\mathit{SCipher}[\tau]$ or $\mathit{ACipher}[\tau]$, encrypt a random element in $[\![\tau]\!]_\eta$ with a fresh random key of the appropriate kind.

Interpreting terms and frames. Given η, we associate with each frame $\varphi = \nu\widetilde{n}.\{x_1 = T_1, \ldots, x_n = T_n\}$ a distribution $[\![\varphi]\!]_\eta$ defined by the following procedure for computing a sample $\phi \xleftarrow{R} [\![\varphi]\!]_\eta$:

1. for each name of sort τ that occurs in φ, draw a value $\widehat{a} \xleftarrow{R} [\![\tau]\!]_\eta$;
2. compute the value $\widehat{T_i}$ of each closed term T_i, recursively:

$$\text{for every function symbol } f, \quad \widehat{f(S_1, \ldots, S_n)} = [\![f]\!]_\eta(\widehat{S_1}, \ldots, \widehat{S_1})$$

3. let the resulting *concrete frame* be $\phi = \{x_1 = \widehat{T_1}, \ldots, x_n = \widehat{T_n}\}$.

We define the notation $[\![\_]\!]_\eta$ for closed terms and tuples of closed terms similarly. We may write $[\![\varphi]\!]_{\eta,\, a_1 \mapsto e_1,\, \ldots,\, a_n \mapsto e_n}$ so as to specify the values for names, and $[\![\varphi]\!]_{\eta,\, \mathsf{c}_1 \mapsto e_1,\, \ldots,\, \mathsf{c}_n \mapsto e_n}$ so as to specify the values of the constants $\mathsf{c}_1, \ldots, \mathsf{c}_n$. We write $[\![\varphi]\!]$ for the *ensemble* (family of distributions) $([\![\varphi]\!]_\eta)_\eta$. We identify a single-valued (Dirac) distribution with its unique value.

Indistinguishability. Two ensembles $D^1 = (D^1_\eta)_\eta$ and $D^2 = (D^2_\eta)_\eta$ are *indistinguishable*, written $D^1 \approx D^2$, when, for every PPTIME adversary A, the function

$$\mathrm{Adv}_A(\eta) = \mathbb{P}\left[e \xleftarrow{R} D^1_\eta : A(e) = 1\right] - \mathbb{P}\left[e \xleftarrow{R} D^2_\eta : A(e) = 1\right]$$

is negligible.

3.2 Cryptographic Assumptions

We use symmetric and asymmetric encryption schemes that satisfy a notion of security related to type-0 and type-1 security [6]. Essentially, we require that for each type τ, the encryption function restricted to elements of $[\![\tau]\!]$ reveal no information about the key used for encryption and hide all partial information about underlying plaintexts—except for their belonging to the carrier set $[\![\tau]\!]$.

Definition 1. *Let $\Pi^s = (\mathcal{K}^s, \mathcal{E}^s, \mathcal{D}^s)$ be a symmetric encryption scheme. For each security parameter η and type $\tau \in T_{\mathsf{senc}}$, we consider the following experiment, with a two-stage PPTIME adversary $A = (A_1, A_2)$:*

- *a key k is generated via $k \xleftarrow{R} \mathcal{K}^s(\eta)$;*
- *A_1 is provided access to an oracle $\mathcal{E}^s(\cdot, k)$, that is, A_1 may submit messages m to the oracle and receives in return corresponding encryptions $\mathcal{E}^s(m, k)$;*
- *then A_1 outputs a challenge message $m^* \in [\![\tau]\!]_\eta$ together with some state information st;*
- *a bit $b \xleftarrow{R} \{0, 1\}$ is selected at random; if $b = 0$, we let c be a (tagged) encryption of m^* under k, that is, $c \xleftarrow{R}$ "SCipher"$\|\tau\|\mathcal{E}^s(m^*, k)$; otherwise, we let c be a (tagged) encryption of a random element of τ under a random key, that is, $c \xleftarrow{R} [\![SCipher[\tau]]\!]_\eta$;*
- *A_2 is given c and st, and outputs a bit b'.*

The adversary A is successful if $b' = b$. The advantage of A is defined by $\mathrm{Adv}^{\tau}_{\Pi^s, A}(\eta) = \Pr[A \text{ is successful}] - \frac{1}{2}$. *We say that Π^s is* T_{senc}-secure *if for all PPTIME adversaries A and all $\tau \in T_{\mathsf{senc}}$, the function $\mathrm{Adv}^{\tau}_{\Pi^s, A}(\cdot)$ is negligible.*

Definition 2. *Let $\Pi^p = (\mathcal{K}^p, \mathcal{E}^p, \mathcal{D}^p)$ be an asymmetric encryption scheme. For each security parameter η and type $\tau \in T_{\mathsf{penc}}$, we consider the following experiment, with a two-stage PPTIME adversary $A = (A_1, A_2)$:*

- *a pair of encryption/decryption keys (pk, sk) is generated via $(pk, sk) \xleftarrow{R} \mathcal{K}^p(\eta)$, and A_1 is given pk;*
- *A_1 outputs a challenge message $m^* \in [\![\tau]\!]_\eta$ together with some state information st;*
- *a bit $b \xleftarrow{R} \{0, 1\}$ is selected at random; if $b = 0$, we let c be a (tagged) encryption of m^* under pk, that is, $c \xleftarrow{R}$ "ACipher"$\|\tau\|\mathcal{E}^p(m^*, pk)$; otherwise, we let c be a (tagged) encryption of a random element of τ under a random public key, that is, $c \xleftarrow{R} [\![ACipher[\tau]]\!]_\eta$;*
- *A_2 is given c and st, and outputs a bit b'.*

The adversary $A = (A_1, A_2)$ is successful if $b' = b$. The advantage of A is defined by $\mathrm{Adv}^{\tau}_{\Pi^p, A}(\eta) = \Pr[A \text{ is successful}] - \frac{1}{2}$. *We say that Π^p is* T_{penc}-secure *if for all PPTIME adversaries A and all $\tau \in T_{\mathsf{penc}}$, the function $\mathrm{Adv}^{\tau}_{\Pi^p, A}(\cdot)$ is negligible.*

Our notion of security for encryption schemes that use data values (such as passwords) as keys is less standard—and there is not yet a standard notion in the area:

Definition 3. *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric, deterministic, type-preserving encryption scheme such that the set of keys is $\{0,1\}^{\alpha_1(\eta)}$ for each η.*

1. **Real-or-Random security (T_{enc}-RoR):** *For each security parameter η and type $\tau \in T_{\mathsf{enc}}$, we consider the following experiment, with a two-stage PPTIME adversary $A = (A_1, A_2)$:*
   - *a key k is generated via $k \xleftarrow{R} \mathcal{K}(\eta)$;*
   - *A_1 is provided access to an oracle $\mathcal{E}(\cdot, k)$, that is, A_1 may submit (tagged) messages m to the oracle and receives in return corresponding (tagged) encryptions $\mathcal{E}(m, k)$;*
   - *then A_1 submits a challenge message $m^* \in [\![\tau]\!]_\eta$ and some state information st;*
   - *a bit $b \xleftarrow{R} \{0,1\}$ is selected at random; if $b = 0$, we let c be the (tagged) encryption of m^* under k, that is, $c = \mathcal{E}(m^*, k)$; otherwise, we let $c \xleftarrow{R} [\![\tau]\!]_\eta$ be a random element from $[\![\tau]\!]_\eta$;*
   - *A_2 is given c and st, and outputs a bit b'.*

   The adversary A is successful if $b' = b$ and *the challenge message m^* is different from all the messages m submitted by A to the encryption oracle. The advantage of A is $\mathrm{Adv}^{\tau}_{RoR,\Pi,A}(\eta) = \Pr[A \text{ is successful}] - \frac{1}{2}$. We say that Π is T_{enc}*-RoR secure *if for all PPTIME adversaries A and all $\tau \in T_{\mathsf{enc}}$, the function $\mathrm{Adv}^{\tau}_{RoR,\Pi,A}(\cdot)$ is negligible.*
2. **Encryption under passwords or other data values (T_{enc}-Pwd):** *For each security parameter η and type $\tau \in T_{\mathsf{enc}}$, we consider the following experiment, with a two-stage PPTIME adversary $A = (A_1, A_2)$:*
   - *A_1 outputs a key $k \in \{0,1\}^{\alpha_1(\eta)}$ and some state information st;*
   - *a bit $b \xleftarrow{R} \{0,1\}$ is selected at random; if $b = 0$, we let c be the (tagged) encryption of some random element under k, that is, $m \xleftarrow{R} [\![\tau]\!]_\eta$ and $c = \mathcal{E}(m, k)$; otherwise, we let $c \xleftarrow{R} [\![\tau]\!]_\eta$ be a random element from $[\![\tau]\!]_\eta$;*
   - *A_2 is given c and st, and outputs a bit b'.*

   The adversary A is successful if $b' = b$. The advantage of A is defined by $\mathrm{Adv}^{\tau}_{Pwd,\Pi,A}(\eta) = \Pr[A \text{ is successful}] - \frac{1}{2}$. We say that Π is a T_{enc}-Pwd secure *if for all PPTIME adversaries A and all $\tau \in T_{\mathsf{enc}}$, the function $\mathrm{Adv}^{\tau}_{Pwd,\Pi,A}(\cdot)$ is negligible.*

Finally, Π is T_{enc}-secure *if it is both T_{enc}-RoR and T_{enc}-Pwd secure.*

Condition 1 (T_{enc}-RoR security) is a variant of IND-P1-C0 security [30, 11]. We require it because we allow enc to be used as a first-class encryption algorithm, that is, with strong keys (not just passwords). Without this condition, our main result remains true on frames which use only constants as keys for enc (much as in [7]).

Condition 2 (T_{enc}-Pwd security) addresses the security of passwords (or other data) when used as keys. Intuitively, it states that the encryption of a random value must be distributed like the value. A related previous condition [7] allows a possibly different distribution for the encryptions of random values and the values themselves. This difference is mostly due to the fact that we authorize multiple layers of encryptions with passwords (see example (4)).

Finally, an implementation with (Π^s, Π^p, Π) is $(T_{\mathsf{senc}}, T_{\mathsf{penc}}, T_{\mathsf{enc}})$-*secure* (or simply *secure*) if the three schemes Π^s, Π^p, and Π are, respectively, T_{senc}-secure, T_{penc}-secure, and T_{enc}-secure.

A possible secure implementation, using standard cryptographic tools, is outlined in the extended version of this paper [1].

4 Soundness of Static Equivalence

In this section we present our main soundness result. As usual (following [6]), this result requires a hypothesis that excludes encryption cycles, and also some other well-formedness conditions.

A *key position* in an expression is a position that corresponds to the argument U of a subterm of the form $\mathsf{pub}(U)$, or to the second argument V of a subterm $\mathsf{enc}_\tau(U, V)$, $\mathsf{penc}_\tau(U, V, W)$, or $\mathsf{senc}_\tau(U, V, W)$. An *encryption cycle* of a frame φ is a sequence of names k_0, k_1, ..., k_n of sort *Data*, *DKey*, and *SKey* such that $k_n = k_0$ and

> for each $0 \leq i \leq n-1$, there exists a subterm of φ of the form $\mathsf{enc}_\tau(U, V)$, $\mathsf{penc}_\tau(U, V, W)$, or $\mathsf{senc}_\tau(U, V, W)$ such that k_i is a subterm of U *not* in key position and k_{i+1} is a subterm of V.

For instance, the frame $\varphi_1 = \{x = \{sk_1\}_{k_2}^{r_1},\ y = \{k_2\}_{\mathsf{pub}(sk_1)}^{r_2}\}$ has an encryption cycle, while $\varphi_2 = \{x = \{\mathsf{pub}(sk_1)\}_{k_2}^{r_1},\ y = \{k_2\}_{\mathsf{pub}(sk_1)}^{r_2}\}$ does not.

A frame φ is *well-formed* if it satisfies the following conditions:

(i) φ is $\mathcal{R}$-reduced, that is, in normal form with respect to the rewriting system $\mathcal{R}$;
(ii) φ does not contain the symbols dec, pdec, sdec, $\mathsf{pdec\_success}$, $\mathsf{sdec\_success}$, fst, and snd;
(iii) terms in key position in φ are of the following forms, depending on their sort:
  - sorts *DKey* and *SKey*: names,
  - sort *EKey*: names and terms of the form $\mathsf{pub}(a)$,
  - sort *Data*: names and constants;
(iv) terms of type *Coins* may only be names, and appear as the third argument of an encryption; moreover, if such a name appears twice in φ then the encryption terms in which it appears are identical;
(v) φ has no encryption cycles;
(vi) for every subterm of φ of the form $\mathsf{enc}(T, k)$ where k is a name, T contains none of the constants w, c_0, c_1, ..., and T has no subterm of the form $\mathsf{enc}(S, 0)$ or $\mathsf{enc}(S, 1)$.

Condition (ii) indicates that we focus on the indistinguishability of expressions built from constructors; it does not preclude using other functions in the observations that may distinguish frames. Condition (iii) says that keys are atomic terms for symmetric encryptions, and terms of the form $\mathsf{pub}(a)$ for public-key encryptions. Similarly, condition (iv) says that coins are names and are used only for encryptions, with different coins in each encryption. Condition (v) is the acyclicity requirement. Finally, condition (vi) restricts the occurrences of constants within plaintexts for deterministic encryption under strong keys (represented by names). For instance, this condition excludes the frame $\nu k.\{x_1 = \mathsf{enc}(\mathsf{c}_1, k),\ x_2 = \mathsf{enc}(\mathsf{c}_2, k)\}$, which is equivalent to $\nu a_1, a_2.\{x_1 = a_1,\ x_2 = a_2\}$ formally but not computationally if c_1 and c_2 happen to have the same bit-string implementations. More generally, when T_1 and T_2 are two terms such that $T_1 \neq_E T_2$, the encryptions $\mathsf{enc}(T_1, k)$ and $\mathsf{enc}(T_2, k)$ may behave like distinct fresh names formally but not computationally, unless the bit-string values of T_1 and T_2 collide with negligible probability.

We obtain:

Theorem 1 ($\approx_E$-soundness). *Let φ_1 and φ_2 be two well-formed frames such that $\varphi_1 \approx_E \varphi_2$. In any secure implementation, $[\![\varphi_1]\!] \approx [\![\varphi_2]\!]$.*

The proof of this theorem (in the extended version of this paper [1]) relies on a detailed formal analysis of static equivalence, and in particular on a decision procedure for the static equivalences under consideration. The theorem follows from a step-by-step complexity-theoretic validation of the decision procedure.

5 Application to Security against Guessing Attacks

Weak secrets such as PINs and passwords sometimes serve as encryption keys. Their safe use is challenging because of the possibility of guessing attacks, in which data that depends on a weak secret allows an attacker to check guesses of the values of the weak secret. For example, if a message contains a fixed cleartext Hello, and it is encrypted under a password pwd drawn from a small dictionary, then an attacker that sees the message can try to decrypt it with all values in the dictionary until one yields the cleartext Hello, thus discovering a probable value for the password. The attacker may mount this attack off-line, avoiding detection. The attack is made possible by the fact that, given the data available to the attacker, pwd can be distinguished from another value pwd': $\mathsf{enc}_{\mathsf{string}}(\mathsf{Hello}, \mathsf{pwd}) \not\approx_E \mathsf{enc}_{\mathsf{string}}(\mathsf{Hello}, \mathsf{pwd'})$. Conversely, immunity to such guessing attacks can be formulated as a static equivalence between two frames, one that corresponds to what is actually available to the attacker and the other to a variant in which the weak secrets are replaced with fresh keys or with arbitrary other keys [17, 20].

We believe that, as suggested in the introduction, the treatment of guessing attacks in terms of static equivalence is attractive in several respects. This section shows that this treatment can be computationally sound. In comparison with the only previous computational justification for a formal criterion against guessing attacks [7], the present results have several strengths. First, they apply to a criterion formulated in terms of standard notions, rather than an ad hoc criterion. Consequently, they fit into a standard analysis method which can also deal with other properties and other kinds of attacks. In addition, they are more general, in that they immediately apply to scenarios with multiple weak secrets. Finally, it is satisfying that these results follow from theorems of somewhat broader interest.

In our formalism, modeling a password as a constant w of sort *Data*, we may say that the password is not revealed by a frame φ if $\varphi\{\mathsf{w} \mapsto \mathsf{c}_0\} \approx_E \varphi\{\mathsf{w} \mapsto \mathsf{c}_1\}$. The substitutions $\{\mathsf{w} \mapsto \mathsf{c}_0\}$ and $\{\mathsf{w} \mapsto \mathsf{c}_1\}$ correspond to instantiations of the password with distinct actual values; each of the frames represents what an attacker may obtain in the course of a protocol execution and then analyze off-line. The soundness of this formal notion is a corollary of Theorem 1, as is a generalization to multiple passwords. The formal notion can be applied to some examples from the literature (such as the EKE protocol [13, 7]), and the corollaries then yield computational guarantees for those examples.

Corollary 1 (Single password). *Assume a secure implementation. Let φ be a well-formed frame, let w be a constant of sort Data, and let c_0, c_1 be two fresh, distinct constants of sort Data. If $\varphi\{\mathsf{w} \mapsto \mathsf{c}_0\} \approx_E \varphi\{\mathsf{w} \mapsto \mathsf{c}_1\}$ then w is computationally hidden in φ: for all PPTIME-computable sequences of bit-strings κ_0, κ_1 with $\kappa_i(\eta) \in \{0,1\}^{\alpha_1(\eta)}$,*

$$[\![\varphi]\!]_{\eta,\mathsf{w}\mapsto\kappa_0(\eta)} \approx [\![\varphi]\!]_{\eta,\mathsf{w}\mapsto\kappa_1(\eta)}$$

Corollary 2 (Multiple passwords). *Assume a secure implementation. Let φ be a well-formed frame, let $\mathsf{w}_1, \ldots, \mathsf{w}_n$ be n constants of sort Data, and let c_0, $\mathsf{c}_1, \ldots, \mathsf{c}_n$ be $n{+}1$ fresh, distinct constants of sort Data. If $\varphi\{\mathsf{w}_1 \mapsto \mathsf{c}_1, \ldots, \mathsf{w}_n \mapsto c_n\} \approx_E \varphi\{\mathsf{w}_1 \mapsto \mathsf{c}_0, \ldots, \mathsf{w}_n \mapsto c_0\}$ then $\mathsf{w}_1, \ldots, \mathsf{w}_n$ are computationally hidden in φ: for all (not necessarily pairwise distinct) PPTIME-computable sequences of bit-strings $\kappa_1 \ldots \kappa_n$, $\kappa'_1 \ldots \kappa'_n$ with $\kappa_i(\eta), \kappa'_i(\eta) \in \{0,1\}^{\alpha_1(\eta)}$,*

$$[\![\varphi]\!]_{\eta,\mathsf{w}_1\mapsto\kappa_1(\eta),\ldots,\mathsf{w}_n\mapsto\kappa_n(\eta)} \approx [\![\varphi]\!]_{\eta,\mathsf{w}_1\mapsto\kappa'_1(\eta),\ldots,\mathsf{w}_n\mapsto\kappa'_n(\eta)}$$

6 Conclusion

In this paper we investigate the computational foundations of a formal notion of data indistinguishability, static equivalence. We define a particular equational theory for which we can obtain a computational soundness result. Although they are largely based on ideas common in previous work, neither the equational theory nor our computational assumptions are straightforward. The main difficulties that we address relate to encryption under data values. Correspondingly, we obtain a soundness result for a formal criterion of protection against guessing attacks on those data values.

A direction for further work is the generalization of our results to other cryptographic primitives. For instance, certain password-based protocols make a sophisticated use of exponentiation, which we do not include in our equational theory. Yet other primitives, such as digital signatures, are important for trace properties and for process equivalences (more so than for static equivalences). We hope that, perhaps with these extensions, the present work may serve as a component of an eventual computational justification of process equivalences.

Acknowledgments. We thank Steve Kremer and the anonymous referees for helpful comments. This research was partly carried out while Mathieu Baudet was visiting the University of California at Santa Cruz and Bogdan Warinschi was at Stanford University. It was partly supported by the National Science Foundation under Grants CCR-0204162, CCR-0208800, CCF-0524078, and ITR-0430594, and by the ARA SSIA Formacrypt and ACI Jeunes Chercheurs JC9005.

References

1. M. Abadi, M. Baudet, and B. Warinschi. Guessing attack and the computational soundness of static equivalence (extended version). Manuscript, 2006.
2. M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. In *Proc. 31st International Colloquium on Automata, Languages and Programming (ICALP'04)*, volume 3142 of *LNCS*, pages 46–58. Springer, 2004.
3. M. Abadi and V. Cortier. Deciding knowledge in security protocols under (many more) equational theories. In *Proc. 18th IEEE Computer Security Foundations Workshop (CSFW'05)*, pages 62–76, 2005.
4. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. 28th ACM Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115, 2001.
5. M. Abadi and A. D. Gordon. A bisimulation method for cryptographic protocols. *Nordic Journal of Computing*, 5(4):267–303, 1998.
6. M. Abadi and P. Rogaway. Reconciling two views of cryptography (The computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, 2002.
7. M. Abadi and B. Warinschi. Password-based encryption analyzed. In *Proc. 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *LNCS*, pages 664–676. Springer, 2005.
8. M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations. In *Proc. 10th ACM Conference on Computer and Communications Security (CCS'03)*, pages 220–330, 2003.
9. G. Bana. *Soundness and completeness of formal logics of symmetric encryption.* PhD thesis, University of Pensilvania, 2004.
10. M. Baudet. Deciding security of protocols against off-line guessing attacks. In *Proc. 12th ACM Conference on Computer and Communications Security (CCS'05)*, pages 16–25, Alexandria, Virginia, USA, Nov. 2005.
11. M. Baudet, V. Cortier, and S. Kremer. Computationally sound implementations of equational theories against passive adversaries. In *Proc. 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *LNCS*, pages 652–663. Springer, 2005.
12. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *Advances in Cryptology – EUROCRYPT'00*, volume 1807 of *LNCS*, pages 139–155. Springer, 2000.
13. S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proc. 1992 IEEE Symposium on Security and Privacy (SSP'92)*, pages 72–84, 1992.
14. B. Blanchet, M. Abadi, and C. Fournet. Automated verification of selected equivalences for security protocols. In *Proc. 20th IEEE Symposium on Logic in Computer Science (LICS'05)*, pages 331–340, 2005.

15. M. Boreale, R. De Nicola, and R. Pugliese. Proof techniques for cryptographic processes. In *Proc. 14th IEEE Symposium on Logic in Computer Science (LICS'99)*, pages 157–166, 1999.
16. V. Boyko, P. MacKenzie, and S. Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In *Advances in Cryptology – EUROCRYPT'00*, volume 1807 of *LNCS*, pages 156–171. Springer, 2000.
17. R. Corin, J. M. Doumen, and S. Etalle. Analysing password protocol security against off-line dictionary attacks. Technical report TR-CTIT-03-52, Centre for Telematics and Information Technology, Univ. of Twente, The Netherlands, 2003.
18. R. Corin, S. Malladi, J. Alves-Foss, and S. Etalle. Guess what? Here is a new tool that finds some new guessing attacks (extended abstract). In *IFIP WG 1.7 and ACM SIGPLAN Workshop on Issues in the Theory of Security (WITS'03)*, pages 62–71, 2003.
19. S. Delaune and F. Jacquemard. A theory of dictionary attacks and its complexity. In *Proc. 17th IEEE Computer Security Foundations Workshop (CSFW'04)*, pages 2–15, 2004.
20. C. Fournet. Private communication, 2002.
21. R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. In *Advances in Cryptology – EUROCRYPT'03*, volume 2656 of *LNCS*, pages 524–543. Springer, 2003.
22. O. Goldreich and Y. Lindell. Session key generation using human passwords only. In *Advances in Cryptology – CRYPTO'01*, volume 2139 of *LNCS*, pages 403–432. Springer, 2001.
23. L. Gong. Verifiable-text attacks in cryptographic protocols. In *Proc. 9th IEEE Conference on Computer Communications (INFOCOM'90)*, pages 686–693, 1990.
24. L. Gong, T. M. A. Lomas, R. M. Needham, and J. H. Saltzer. Protecting poorly chosen secrets from guessing attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):648–656, 1993.
25. J. Katz, R. Ostrovsky, and M. Yung. Practical password-authenticated key exchange provably secure under standard assumptions. In *Advances in Cryptology – EUROCRYPT'01*, volume 2045 of *LNCS*, pages 475–494. Springer, 2001.
26. P. Laud. Symmetric encryption in automatic analyses for confidentiality against active adversaries. In *Proc. 2004 IEEE Symposium on Security and Privacy (SSP'04)*, pages 71–85, 2004.
27. G. Lowe. Analysing protocols subject to guessing attacks. *Journal of Computer Security*, 12(1):83–98, 2004.
28. D. Micciancio and B. Warinschi. Completeness theorems for the Abadi-Rogaway logic of encrypted expressions. *Journal of Computer Security*, 12(1):99–129, 2004.
29. D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In *Proc. Theory of Cryptography Conference (TCC'04)*, volume 2951 of *LNCS*, pages 133–151. Springer, 2004.
30. D. H. Phan and D. Pointcheval. About the security of ciphers (semantic security and pseudo-random permutations). In *Proc. Selected Areas in Cryptography (SAC'04)*, volume 3357 of *LNCS*, pages 185–200. Springer, 2004.
31. M. Steiner, G. Tsudik, and M. Waidner. Refinement and extension of encrypted key exchange. *ACM SIGOPS Oper. Syst. Rev.*, 29(3):22–30, 1995.
32. G. Tsudik and E. V. Herreweghen. Some remarks on protecting weak secrets and poorly-chosen keys from guessing attacks. In *Proc. 12th IEEE Symposium on Reliable Distributed Systems (SRDS'93)*, 1993.