# Password-Based Encryption Analyzed

Martín Abadi[1] and Bogdan Warinschi[2]

[1] Computer Science Department, University of California, Santa Cruz
[2] Computer Science Department, Stanford University

**Abstract.** The use of passwords in security protocols is particularly delicate because of the possibility of off-line guessing attacks. We study password-based protocols in the context of a recent line of research that aims to justify symbolic models in terms of more concrete, computational ones. We offer two models for reasoning about the concurrent use of symmetric, asymmetric, and password-based encryption in protocol messages. In each of the models we define a notion of equivalence between messages and also characterize when passwords are used securely in a message or in a set of messages. Our new definition for the computational security of password-based encryption may be of independent interest. The main results of this paper are two soundness theorems. We show that under certain (standard) assumptions about the computational implementation of the cryptographic primitives, symbolic equivalence implies computational equivalence. More importantly, we prove that symbolically secure uses of passwords are also computationally secure.

## 1 Introduction

Passwords and other weak secrets sometimes serve as cryptographic keys in security protocols and elsewhere (e.g., [5, 14, 16, 21]). The use of weak secrets is particularly delicate because of the possibility of off-line guessing attacks. In such an attack, data that depends on a weak secret is used in checking guesses of the values of the weak secret. Consider, for example, a protocol where two parties exchange the encryption $c$ of some fixed message, say $\mathsf{Ok}$, under a shared password $pwd$. If $pwd$ is picked from a relatively small dictionary, then an attacker that obtains a transcript of the protocol execution can mount the following off-line attack. It decrypts the ciphertext $c$ with the passwords in the dictionary, one by one, until the result of the decryption is the text $\mathsf{Ok}$. The password used for this last decryption is likely to be $pwd$. Guessing attacks such as this one are passive, in the sense that they do not require interaction with the protocol participants, so they are hard to detect. A guessing attack may however be carried out after an active attack, relying on the messages exchanged in the course of the active attack.

Early research on the design and analysis of protocols based on weak secrets focused on techniques for defending against guessing attacks (e.g. [13]). These techniques basically aim to ensure that plaintexts encrypted under passwords do not contain redundancy that can later be used to verify a password guess. While this is a helpful guideline, its informal application need not guarantee security. As experience demonstrates (e.g., [23]), conjecturing the security of a protocol, or arguing it only heuristically, is not sufficient. Instead, and this is the goal pursued by recent research on the subject, the security of protocols should be rigorously analyzed. Models for carrying out such analyzes have been designed using two different, yet related approaches.

The first approach, known as the symbolic or formal methods approach, adopts an abstract view of executions. Messages are modeled as elements of a term algebra constructed with symbolic operations that represent various cryptographic primitives. Parties operate on terms using a limited number of inference rules, sometimes generically known as the Dolev-Yao rules. The rules reflect a common understanding of the security of cryptographic primitives. For example, they say that the message encrypted in a ciphertext can be recovered only if the appropriate decryption key is known. Quite often, proofs that rely on these rules can be mechanized. Work done on symbolic models for password-based protocol has concentrated on extending the Dolev-Yao rules to guessing attacks [7–9, 18]. Typical formalisms enrich standard symbolic models with an operation that represents encryption under passwords, and they offer a careful account of when a password guess can be verified from a given set of terms (presumably a transcript of a protocol execution). The resulting decision procedure has been automated [7, 8, 18]; a corresponding decision problem has been shown NP-complete [9]. Unfortunately, as remarked by authors of prior work [18], it is quite difficult to determine if a set of formal criteria for the existence of guessing attacks is exhaustive. Hence, the possibility remains that a formal analysis would miss some attacks, and unsoundly conclude that a protocol is secure when in fact it is not.

The second approach, known as the computational approach, uses a concrete (bit level) representation, for protocol executions. The attacker is modeled as a powerful, arbitrary probabilistic polynomial-time Turing machine. Although proofs with this approach tend to be lengthy, difficult, and tedious, it is generally accepted that it provides strong guarantees. For the case of password-based protocols, work with the computational approach seems to have focused almost exclusively on the important use of passwords for authenticated key exchange. This work includes designing models and giving provably secure constructions [4, 6, 10, 11, 15]. Surprisingly, the security of password-based encryption as a stand-alone primitive has not been addressed.

A recent line of research aims to justify the abstractions made by symbolic methods with respect to computational models (e.g., [1, 2, 17, 20, 22]) via soundness theorems. These theorems typically state that, under certain assumptions on the implementation of cryptographic primitives, symbolic security proofs imply security in the computational model. The applications of soundness theorems are quite appealing: simple reasoning techniques and automatic tools, specific to the symbolic setting, can be used to carry out proofs that guarantee strong, computational security.

This paper is a first exploration on the subject of computationally sound symbolic analysis for protocols based on passwords. We concentrate on off-line guessing attacks, because they are the main original concern in the analysis of those protocols and because they appear mostly orthogonal to the standard active attacks. Our framework is an extension of the framework introduced by Abadi and Rogaway [1] to asymmetric and password-based encryption. That framework focuses, as an initial step, on passive attacks; in that respect, it is a good match for our purposes, since off-line guessing attacks are passive by definition (even if they may occur in conjunction with active attacks).

We introduce a language of expressions whose elements are abstract representations of the messages sent during protocol executions. The cryptographic primitives that we consider are symmetric encryption, asymmetric encryption, and encryption that uses passwords as keys. To these expressions we attach two different semantics. The first is symbolic; it is based on an extension of the classical Dolev-Yao inference rules to include password-based encryption. The second is computational; it is based on concrete implementations of the encryption operations. In current protocols, password-based encryption typically serves for achieving authenticity rather than secrecy properties, despite the use of the term "encryption" (which we preserve for historical reasons). Accordingly, our semantics do not require that the encryption of a plaintext under a password ensure the secrecy of the plaintext. Further, we give symbolic and computational definitions for expression equivalence (when two expressions convey the same information to an adversary) and for secure use of passwords (which expressions do not leak passwords despite guessing attacks). The main results of our paper are soundness theorems that link the two models. We prove that if two expressions are equivalent symbolically then they are equivalent computationally. We also prove that if an expression hides a password symbolically then it hides the password computationally.

In Section 2 we give the syntax of the language of expressions. As a counterpart, we introduce a computational setting in Section 4. We define expression equivalence and password hiding, symbolically and computationally, in Sections 3 and 5, respectively. In Section 6 we give our main results; as an example, we show an application to the $\mathsf{EKE}$ protocol [5]. We conclude in Section 7. Because of space constraints, we leave many details and proofs to a longer version of this paper.

## 2 Syntax

In this section we define the language of expressions $\mathsf{Exp}$. We consider messages constructed from bits and cryptographic keys by using pairing, symmetric and asymmetric encryption, as well as encryption that employs passwords as keys. In what follows, $\mathsf{Bool}$ is the set of bits $\{0, 1\}$. $\mathsf{Keys}$ is the set of cryptographic keys; it is the union of the disjoint sets $\mathsf{SKeys}$, $\mathsf{EKeys}$, $\mathsf{DKeys}$, and $\mathsf{Passwd}$ which contain symbols for symmetric keys, asymmetric encryption keys, asymmetric decryption keys, and passwords, respectively. We write $\mathsf{EncKeys}$ for $\mathsf{SKeys} \cup \mathsf{EKeys} \cup \mathsf{Passwd}$, the set of keys that can be used for encryption; and write $(\cdot)^{-1} : \mathsf{SKeys} \cup \mathsf{EKeys} \rightarrow \mathsf{SKeys} \cup \mathsf{DKeys}$ for a bijection that maps an encryption key to the associated decryption key. We usually follow the convention that $K_1^s, K_2^s, \ldots$ represent symmetric keys, $K_1^e, K_2^e, \ldots$ asymmetric encryption keys, and $K_1^d, K_2^d, \ldots$ the corresponding asymmetric decryption keys. In this paper we concentrate on the simple setting where expressions use a single password symbol for encryption, so the set $\mathsf{Passwd}$ contains a single element $W$. This setting is sufficient for analyzing multiple concurrent runs of the execution of a protocol between principals that share a password; with some complications, our approach extends to the general case where multiple passwords are used simultaneously.

The set $\mathsf{Exp}$ of formal expressions is defined by the grammar:

$$\mathsf{Exp} ::= \mathsf{Bool} \mid \mathsf{Passwd} \mid \mathsf{EKeys} \mid \mathsf{DKeys} \mid \mathsf{SKeys} \mid (\mathsf{Exp}, \mathsf{Exp}) \mid \{\mathsf{Exp}\}_{\mathsf{EncKeys}}$$

For example, expression $\{K^s\}_W, \{(0,0)\}_{K^s}$ represents the encryption of symmetric key $K^s$ under the password $W$ paired with the encryption of $(0,0)$ under key $K^s$. The expression $\{K^e\}_{K^e}, \{K^s\}_{K^e}, \{0\}_{K^s}$ represents the encryption of public key $K^e$ under itself, paired with the encryption of symmetric key $K^s$ under $K^e$ and the encryption of the bit $0$ under $K^s$. As we do here, we omit parenthesis when there is no risk of ambiguity or when ambiguity is harmless.

An important subset of $\mathsf{Exp}$ is that of *acyclic expressions*. Acyclicity was introduced in previous work [1] for expressions that use only symmetric encryption. Here we generalize this notion to deal also with asymmetric encryption. Given an expression $E \in \mathsf{Exp}$ we build the following directed graph. The nodes of the graph are pairs of encryption and decryption keys $(K, K^{-1}) \in \mathsf{SKeys} \times \mathsf{SKeys} \cup \mathsf{EKeys} \times \mathsf{DKeys}$ for which at least one of the components appears in $E$. We add an edge between nodes $(K_1, K_1^{-1})$ and $(K_2, K_2^{-1})$, and say that $K_1$ encrypts $K_2$, if there exists $E' \in \mathsf{Exp}$ such that $\{E'\}_{K_1}$ occurs in $E$ and $K_2^{-1}$ occurs in $E'$. We say that an expression $E \in \mathsf{Exp}$ is acyclic if its associated graph is acyclic. For example, the two expressions $\{K_1^e\}_{K_1^e}$ and $(\{K_1^e\}_{K_1^s}, \{K_1^s\}_{K_1^e})$ are acyclic, and the three expressions $\{K_1^s\}_{K_1^s}$, $(\{K_1^s\}_{K_2^s}, \{K_2^s\}_{K_1^s})$, and $(\{K_1^d\}_{K_1^s}, \{K_1^s\}_{K_1^e})$ are not.

## 3 A symbolic model for expressions

In this section we introduce a symbolic semantics for expressions in $\mathsf{Exp}$. Intuitively, the semantics of an expression is a *pattern* that represents the information that an adversary learns by observing the expression. With this interpretation, we give an equivalence relation on the set of expressions that identifies expressions that convey the same information to the adversary (extending [1]). Furthermore, we use the symbolic semantics to give a characterization of expressions that do not leak a password.

*Symbolic semantics and expression equivalence.* The inference rules that an adversary can use for deriving new information are formalized by the *entailment* relation $M \vdash N$, which is the least relation that satisfies:

1. $M \vdash 0$ and $M \vdash 1$,
2. $M \vdash M$,
3. if $M \vdash N_1$ and $M \vdash N_2$ then $M \vdash (N_1, N_2)$,
4. if $M \vdash (N_1, N_2)$ then $M \vdash N_1$ and $M \vdash N_2$,
5. if $M \vdash N$ and $M \vdash K$ then $M \vdash \{N\}_K$, for $K \in \mathsf{EncKeys}$,
6. if $M \vdash \{N\}_K$ and $M \vdash K^{-1}$ then $M \vdash N$, for $K \in \mathsf{SKeys} \cup \mathsf{EKeys}$,
7. if $M \vdash \{N\}_W$ then $M \vdash N$, for $W \in \mathsf{Passwd}$.

Most of the rules are self-explanatory: for example, they allow an adversary to pair messages that it knows (rule (3)), recover the components of a pair (rule (4)), and compute the encryption of a message $M$ under a certain key $K$, provided the adversary can compute both $M$ and $K$ (rule (5)). Rule (6) is the standard rule of Dolev-Yao deduction systems: an adversary can decrypt a ciphertext if it has the right decryption key. For instance, we have: $K_1^d, \{W\}_{K_1^e} \vdash W$ and $\{\{K_1^s\}_{K_2^s}, \{K_1^d\}_{K_1^s}\}_{K_2^s}, \{W\}_{K_1^e}, K_2^s \vdash W$.

Rule (7) shows that our definitions make a pessimistic (but perhaps realistic!) assumption about the secrecy of plaintexts encrypted under a password. For instance, we have $K_1^d, \{\{0, K_2^s\}_{K_1^e}\}_W \vdash K_2^s$. As indicated in the introduction, this assumption is compatible with current uses of passwords for authentication. However, none of the rules allows the recovery of $W$ by simply observing encryptions of messages under $W$.

Patterns are elements of the language $\mathsf{Pat}$ obtained by extending the language $\mathsf{Exp}$ with symbols that represent undecryptable (symmetric and asymmetric) ciphertexts. We let $\mathsf{Undec} = \{\Box^s, \Box^a\}$. The set of patterns is defined by the grammar:

$$\mathsf{Pat} ::= \mathsf{Exp} \mid \mathsf{Undec} \mid (\mathsf{Pat}, \mathsf{Pat}) \mid \{\mathsf{Pat}\}_{\mathsf{EncKeys}}$$

The pattern $p(M, T)$ represents what an adversary can see in an expression $M \in \mathsf{Exp}$ using for decryption the keys in $T \subseteq \mathsf{Keys}$. It is defined inductively by:

$$\begin{aligned}
p(M,T) &= M \qquad\qquad \text{for } M \in \mathsf{Bool} \cup \mathsf{Keys} \\
p((M,N),T) &= (p(M,T), p(N,T)) \\
p(\{M\}_W, T) &= \{p(M,T)\}_W \\
p(\{M\}_{K^s}, T) &= \begin{cases} \{p(M,T)\}_{K^s} & \text{if } K^s \in T \\ \Box^s & \text{otherwise} \end{cases} \\
p(\{M\}_{K^e}, T) &= \begin{cases} \{p(M,T)\}_{K^e} & \text{if } K^d \in T \\ \Box^a & \text{otherwise} \end{cases}
\end{aligned}$$

We let $\mathit{recoverable}(M) = \{K \in \mathsf{Keys} \mid M \vdash K\}$ be the set of keys that can be recovered from an expression $M$. The pattern associated to $M$ is the pattern computed from $M$ given the set of keys recoverable from $M$, that is: $\mathit{pattern}(M) = p(M, \mathit{recoverable}(M))$. For instance, in the case of the expression $\{K_1^s\}_W, \{K_1^e\}_W, \{\{K_2^s\}_{K_1^s}\}_{K_3^s}, \{0\}_{K_1^e}, K_2^s, \{\{K_1^d\}_{K_3^s}, 0\}_{K_1^s}$, the recoverable keys are $K_1^s$, $K_1^e$, and $K_2^s$, and the pattern is $\{K_1^s\}_W, \{K_1^e\}_W, \Box^s, \Box^a, K_2^s, \{\Box^s, 0\}_{K_1^s}$.

We use patterns for defining equivalence of expressions: two expressions are equivalent if they have the same pattern. Much as in previous work, this equivalence relation can be a little too restrictive, so we relax it by using key renaming functions. A key renaming function is a bijection on the set $\mathsf{Keys}$ that preserves the types of keys: it maps passwords to passwords, asymmetric encryption (decryption) keys to asymmetric encryption (respectively decryption) keys, and symmetric keys to symmetric keys.

**Definition 1.** *$M \equiv N$ if and only if $\mathit{pattern}(M) = \mathit{pattern}(N)$, and $M \cong N$ if and only if there exists a key renaming $\sigma$ such that $M \equiv N\sigma$.*

For example, we have $\{0\}_{K_1^s} \cong \{1\}_{K_2^s}$ and $\{0\}_{K_1^e} \cong \{1\}_{K_2^e}$. These equivalences reflect the standard assumption that symmetric and asymmetric encryption hide plaintexts. We also have $\{0\}_{K^s} \ncong \{0\}_{K^e}$: symmetric and asymmetric ciphertexts can in principle be distinguished. Coming to passwords, we have $\{0\}_W \cong \{0\}_W$ and $\{0\}_W \ncong \{1\}_W$: password-based encryptions of different known plaintexts are inequivalent. On the other hand, we have $\{K_1^s\}_W \cong \{K_2^s\}_W$: encryptions of random keys with a password cannot be distinguished. Finally, in contrast, we have $(\{K_1^s\}_W, \{0\}_{K_1^s}) \ncong (\{K_2^s\}_W, \{1\}_{K_2^s})$: if keys encrypted with a password are used elsewhere, then the two resulting expressions may not be equivalent anymore.

*Secure use of passwords, symbolically.* Next we identify a set of expressions in which a password is used securely, that is, the password is not subject to a guessing attack. Our definition is in two steps. First we introduce patterns with variables. Then we say that an expression uses passwords securely if its pattern can be obtained from a pattern with variables by instantiating the variables in a certain *appropriate* way.

Let $\mathsf{Var} = \{x_1, x_2, \ldots\}$ be a set of variables. The set $\mathsf{Pat}[\mathsf{Var}]$ of patterns with variables from $\mathsf{Var}$ is defined by the grammar:

$$\begin{aligned}\mathsf{Pat}[\mathsf{Var}] ::= {} & \mathsf{Bool} \mid \mathsf{EKeys} \mid \mathsf{DKeys} \mid \mathsf{SKeys} \mid \mathsf{Undec} \mid (\mathsf{Pat}[\mathsf{Var}], \mathsf{Pat}[\mathsf{Var}]) \mid \\ & \{\mathsf{Pat}\}_{\mathsf{EKeys}} \mid \{\mathsf{Pat}\}_{\mathsf{SKeys}} \mid \{\mathsf{Var}\}_{\mathsf{Passwd}}\end{aligned}$$

Informally, in a pattern with variables, a password may appear only as an encryption key, and only be used for encrypting variables. For example, $(\{x_1\}_W, \{(\{x_2\}_W, 0)\}_{K^s})$ is in $\mathsf{Pat}[\mathsf{Var}]$, but $(W, \{x_1\}_W)$ and $\{W\}_{K^s}$ are not. Intuitively, the variables mark places in an expression where we can place concrete subexpressions.

For security, we should ensure that these subexpressions do not offer redundancy that could permit a guessing attack. The subexpressions that we consider benign (in this sense) are ciphertexts and keys that do not themselves appear elsewhere in the pattern. More precisely, an instantiation of a pattern with variables into a pattern is appropriate if variables are mapped to one of the symbols $\Box^s$ or $\Box^a$ or to (symmetric or asymmetric) encryption keys that do not appear elsewhere in the pattern. For example, the pattern with variables $\{x_1\}_W, \{(\{x_2\}_W, 0)\}_{K^s}, K^s$ has occurrences of $K^s$, so it cannot be instantiated to $\{K^s\}_W, \{(\{\Box^s\}_W, 0)\}_{K^s}, K^s$. On the other hand, it can be instantiated to $\{\Box^a\}_W, \{(\{K^e\}_W, 0)\}_{K^s}, K^s$ and to $\{K^e\}_W, \{(\{\Box^s\}_W, 0)\}_{K^s}, K^s$ via the appropriate instantiations $[x_1 \mapsto \Box^a, x_2 \mapsto K^e]$ and $[x_1 \mapsto K^e, x_2 \mapsto \Box^s]$. Hence, we define:

**Definition 2.** *Let $p \in \mathsf{Pat}[\mathsf{Var}]$. A mapping $\sigma : \mathsf{Var} \to \mathsf{Pat}$ is appropriate for $p$ if for all $x \in \mathsf{Var}$ it holds that $\sigma(x) \in \mathsf{SKeys} \cup \mathsf{EKeys} \cup \{\Box^s, \Box^a\}$ and, if $\sigma(x)$ is a key $K \in \mathsf{SKeys} \cup \mathsf{EKeys}$, then neither $K$ nor $K^{-1}$ occur in $p$.*

**Definition 3.** *An expression $E \in \mathsf{Exp}$ hides passwords symbolically if there exist $p \in \mathsf{Pat}[\mathsf{Var}]$ and a mapping $\sigma : \mathsf{Var} \to \mathsf{Pat}$ appropriate for $p$ such that $pattern(E) = p\sigma$.*

For example, the expression $\{\{(0, 1)\}_{K^e}\}_W, \{(\{K^e\}_W, 0)\}_{K^s}, K^s$ hides passwords symbolically: its pattern is $\{\Box^a\}_W, \{(\{K^e\}_W, 0)\}_{K^s}, K^s$ which, as noted above, can be obtained from a pattern with variables via an appropriate instantiation. On the other hand, neither $\{0\}_W$ nor $\{(K^s_1, K^s_2)\}_W$ hide passwords symbolically. The former is subject to the attack we sketched in the introduction. The same attack may apply to the latter if any kind of fixed delimiters are used to implement pairing. This possibility cannot be ruled out *a priori*, and is in fact quite reasonable, so we chose to consider this expression insecure. Further, $(\{K^s\}_W, \{0\}_{K^s})$ does not hide passwords symbolically either. Although $W$ encrypts the symmetric key $K^s$ (potentially a random string), and therefore the same attack does not seem to apply, the key $K^s$ is also used for encrypting a fixed plaintext, which allows a simple guessing attack: an adversary decrypts the first part with a possible password, then uses the result for decrypting the second part in order to check the password guess. It might appear that the same attack does not apply to

$\{K_1^s\}_W, \{K_2^s\}_{K_1^s}$, since here the key $K_1^s$ is used for encrypting another symmetric key. We consider this expression insecure because the symmetric encryption scheme may well provide a mechanism for ensuring that decryptions succeed only if the appropriate key is used, as in the case of authenticated encryption (e.g. [3]), thus offering an indirect way to check a password guess.

Our definitions are at the same level of abstraction as those found in the literature on formal analysis of guessing attacks. However, those tend to be, at least superficially, in a somewhat different style. They also model (symmetric and asymmetric) deterministic encryption, while we focus on probabilistic (symmetric and asymmetric) encryption, because this is the standard kind of encryption used in modern cryptography. We expect that a secure expression in the sense defined in this paper is also secure against the symbolic guessing attacks captured by previous work.

## 4 Computational security of encryption schemes

A password-based encryption scheme $\Pi^p$ is given by a pair of polynomial-time algorithms $(\mathcal{E}^p, \mathcal{D}^p)$ for encryption and decryption, respectively. The scheme is used for encrypting messages in a set $\mathsf{Plaintext}(\Pi^p) \subseteq \{0,1\}^*$ under passwords from a dictionary $\mathsf{D} \subseteq \{0,1\}^*$. The messages may be chosen according to a probability distribution, part of a distribution ensemble (a parameterized family of distributions). Thus, for generality, we partition the set of plaintexts and the set of passwords according to a security parameter: $\mathsf{Plaintext}(\Pi^p) = \cup_\eta \mathsf{Plaintext}(\Pi^p)_\eta$ and $\mathsf{D} = \cup_\eta \mathsf{D}_\eta$. Furthermore, we require that dictionaries can be sampled efficiently: each dictionary $\mathsf{D}$ comes with a probabilistic polynomial-time algorithm that, for security parameter $\eta$, returns a sample $w$ from $\mathsf{D}_\eta$; we write this $w \overset{R}{\leftarrow} \mathsf{D}_\eta$. For each $\eta$, the encryption function takes as input a password $pwd \in \mathsf{D}_\eta$ and a plaintext $m \in \mathsf{Plaintext}(\Pi^p)_\eta$ and returns an encryption $\mathcal{E}^p(pwd, m)$ of $m$ under $pwd$. The decryption function $\mathcal{D}^p$ takes as input a password $pwd$ and a ciphertext $c$ and returns the decryption $\mathcal{D}^p(pwd, c)$ of $c$ using $pwd$. For any security parameter $\eta$, any $m \in \mathsf{Plaintext}_\eta$, and $pwd \in \mathsf{D}_\eta$, the equality $m = \mathcal{D}^p(pwd, \mathcal{E}^p(pwd, m))$ must hold.

Before this work, it appears that the security of password-based encryption had not been defined from a computational perspective. We aim to fill this gap. Our definition captures the idea that, given the encryptions of one or more plaintexts under a password, it should be hard to recover the password—and, as suggested in the introduction, our definition does not capture any possible, additional authenticity or secrecy properties. A common assumption is that passwords are selected from a relatively small dictionary that is likely to be known to an adversary; the attack sketched in the introduction indicates that, unless the plaintexts are selected from a distribution with sufficient entropy, there is no hope for the password to be secure. Therefore, in our definition, the plaintexts are chosen according to distributions. Moreover, the distributions are parameterized by a security parameter; we require that it be hard to recover the password asymptotically.

For instance, let us consider a protocol where two parties have exchanged a session key $k$ (for a security parameter $\eta$), without authentication, and wish to use a shared password $pwd$ for authenticating $k$. For this purpose, one party might encrypt a predefined message, say $\mathsf{Ok}$, under $k$, with a symmetric encryption algorithm $\mathcal{E}^s$, then encrypt it further under $pwd$, and transmit the result $\mathcal{E}^p(pwd, \mathcal{E}^s(k, \mathsf{Ok}))$. The other party would first decrypt the message that it receives using $pwd$ and then $k$. It would accept $k$ as valid only if the result of this last decryption is Ok. Ideally, $\mathcal{E}^s(k, \mathsf{Ok})$ should not expose redundancy, so that it can be safely encrypted under $pwd$. The security of $pwd$ can be guaranteed only for large values of $\eta$: for small values, an adversary that sees $\mathcal{E}^p(pwd, \mathcal{E}^s(k, \mathsf{Ok}))$ can check a password guess $pwd'$ by decrypting with $pwd'$ and then breaking the inner encryption—a feasible task for small values of $\eta$.

The following definition of security uses an adversary $A$ that has access to an encryption oracle $\mathcal{E}^p(pwd, Dist)$. At each query to the oracle, the oracle samples a string $d$ according to distribution $Dist$ and returns $\mathcal{E}^p(pwd, d)$, the encryption of $d$ under $pwd$. Intuitively, the definition says that $A$ cannot tell which of $w_0$ and $w_1$ (two possible values of $pwd$) is used for creating encryptions of plaintexts selected according to $Dist$.

**Definition 4.** *A dictionary $\mathsf{D}^0$ is a subdictionary of $\mathsf{D}$ if $\mathsf{D}^0_\eta \subseteq \mathsf{D}_\eta$ for all $\eta$. A dictionary $\mathsf{D}^0$ is a singleton dictionary if $|\mathsf{D}^0_\eta| = 1$ for all $\eta$. Let $\Pi^p = (\mathcal{E}^p, \mathcal{D}^p)$ be a password-based encryption scheme. We say that $\Pi^p$ securely encrypts distribution ensemble $Dist = (Dist_\eta)_\eta$ using passwords from dictionary $\mathsf{D}$, if for any probabilistic polynomial-time adversary $A$, and any singleton subdictionaries $\mathsf{D}^0$ and $\mathsf{D}^1$ of $\mathsf{D}$,*

$$\begin{aligned}\mathsf{Adv}_{\Pi^p, Dist, A}(\eta) \overset{\text{def}}{=} \Pr\left[w_0 \overset{R}{\leftarrow} \mathsf{D}^0_\eta, w_1 \overset{R}{\leftarrow} \mathsf{D}^1_\eta :\ A^{\mathcal{E}^p(w_1, Dist_\eta)}(\eta, w_0, w_1) = 1\right] - \\ \Pr\left[w_0 \overset{R}{\leftarrow} \mathsf{D}^0_\eta, w_1 \overset{R}{\leftarrow} \mathsf{D}^1_\eta :\ A^{\mathcal{E}^p(w_0, Dist_\eta)}(\eta, w_0, w_1) = 1\right]\end{aligned}$$

*is negligible (as a function of the security parameter $\eta$).*

(Recall that a function is negligible if it is smaller than the inverse of any polynomial for all sufficiently large inputs.)

In defining the syntax of password-based encryption, we do not require that the encryption function be randomized. Interestingly, randomization and security appear to be somewhat in conflict for password-based encryption. In order to explain this observation, let us write $\mathcal{E}^p(pwd, m, r)$ for the encryption of $m$ under password $pwd$ with random coins $r$. Consider an adversary $A$ with access to an encryption oracle as in the definition above, but now with the (reasonable) capability of obtaining several encryptions of the same plaintext using different random coins. When $A$ queries the encryption oracle twice, it obtains ciphertexts $c_0 = \mathcal{E}^p(w_b, m, r_0)$ and $c_1 = \mathcal{E}^p(w_b, m, r_1)$ for some $b \in \{0, 1\}$, some plaintext $m$, and some fresh random coins $r_0$ and $r_1$. Suppose that $b = 0$, without loss of generality. When $A$ decrypts $c_0$ and $c_1$ with $w_0$, it obtains $m$ twice. For $b$ to remain secret, it also must be the case that $\mathcal{D}^p(w_1, \mathcal{E}^p(w_0, m, r_0)) = \mathcal{D}^p(w_1, \mathcal{E}^p(w_0, m, r_1))$. In this sense, the use of the random coins is trivial.

In addition to password-based encryption schemes, we rely on symmetric and asymmetric encryption schemes. As usual, a symmetric or asymmetric encryption scheme consists of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ for key generation, encryption, and decryption. We require that these satisfy a variant of the standard notion of semantic security [12], called *type-0 security*. This notion was previously introduced for the case of symmetric encryption [1] and extends to the case of asymmetric encryption. We leave precise definitions and constructions for a longer version of this paper.

An *encryption suite* is a triple $\mathit{\Pi} = (\mathit{\Pi}^a, \mathit{\Pi}^s, \mathit{\Pi}^p)$ with an asymmetric encryption scheme $\mathit{\Pi}^a$, a symmetric encryption scheme $\mathit{\Pi}^s$, and a password-based encryption scheme $\mathit{\Pi}^p$. We say that an encryption suite is secure if it provides type-0 secure asymmetric and symmetric encryption schemes, and its password-based encryption scheme securely encrypts keys and ciphertexts:

**Definition 5.** *An encryption suite $\mathit{\Pi} = (\mathit{\Pi}^a, \mathit{\Pi}^s, \mathit{\Pi}^p)$ is secure if $\mathit{\Pi}^a = (\mathcal{K}^a, \mathcal{E}^a, \mathcal{D}^a)$ and $\mathit{\Pi}^s = (\mathcal{K}^s, \mathcal{E}^s, \mathcal{D}^s)$ are type-0 secure encryption schemes and $\mathit{\Pi}^p$ securely encrypts distribution ensembles* sym key, sym ciphertext, asym key, *and* asym ciphertext *defined by the algorithms below:*

| sym key($\eta$) | sym ciphertext($\eta$) | asym key($\eta$) | asym ciphertext($\eta$) |
|---|---|---|---|
| $(k,k) \overset{R}{\leftarrow} \mathcal{K}^s(\eta)$ | $(k,k) \overset{R}{\leftarrow} \mathcal{K}^s(\eta)$ | $(pk,sk) \overset{R}{\leftarrow} \mathcal{K}^a(\eta)$ | $(pk,sk) \overset{R}{\leftarrow} \mathcal{K}^a(\eta)$ |
| *As a sample* | *As a sample* | *As a sample* | *As a sample* |
| *return* $k$ | *return* $\mathcal{E}^s(k, \mathbf{0})$ | *return* $pk$ | *return* $\mathcal{E}^a(pk, \mathbf{0})$ |

## 5 A computational model for expressions

In this section we give a computational interpretation to expressions in the form of ensembles of probability distributions, and give computational definitions for expression equivalence and password hiding.

For an encryption suite $\mathit{\Pi}$ and a dictionary $\mathsf{D} = \cup_\eta \mathsf{D}_\eta$, we associate with each expression $M \in \mathsf{Exp}$ a distribution $[\![M]\!]_{\mathit{\Pi}[\eta],\mathsf{D}}$ on strings of bits, and thereby an ensemble $[\![M]\!]_{\mathit{\Pi},\mathsf{D}}$. The definition is inductive:

– Each key symbol $K$ that occurs in $M$ is mapped to a string $\tau(K)$, via the key generation algorithms of $\mathit{\Pi}^s$ and $\mathit{\Pi}^a$ for symmetric and asymmetric keys, respectively, and by selecting at random from $\mathsf{D}_\eta$ for passwords.
– The formal bits 0 and 1 are mapped to standard string representations for them.
– The image of a pair $(M, N)$ is obtained by concatenating the images of $M$ and $N$.
– The image of a formal encryption $\{M\}_K$ is obtained by calculating $\mathcal{E}^{t(K)}_{\tau(K)}(x)$, where $x$ is the image of $M$ and $t(K) \in \{a, s, p\}$ selects the type of encryption.

**Definition 6.** *Two ensembles $D^0$ and $D^1$ are indistinguishable ($D^0 \approx D^1$) if for any probabilistic polynomial-time algorithm $A$,*

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

*is negligible (as a function of the security parameter $\eta$).*

**Definition 7.** *The expressions $E_0, E_1 \in \mathsf{Exp}$ are computationally equivalent if their associated distribution ensembles are indistinguishable, that is, $[\![E_0]\!]_{\mathit{\Pi},\mathsf{D}} \approx [\![E_1]\!]_{\mathit{\Pi},\mathsf{D}}$.*

**Definition 8.** *Let $\mathit{\Pi}$ be an arbitrary encryption suite and let $\mathsf{D}$ be a dictionary. An expression $E \in \mathsf{Exp}$ hides passwords in $\mathsf{D}$ computationally if for all singleton dictionaries $\mathsf{D}^0$ and $\mathsf{D}^1$, subdictionaries of $\mathsf{D}$, it holds that $[\![E]\!]_{\mathit{\Pi},\mathsf{D}^0} \approx [\![E]\!]_{\mathit{\Pi},\mathsf{D}^1}$.*

In this definition, intuitively, an adversary is given two singleton dictionaries and a sample from the distribution associated with the expression $E$. This sample is created by using one of the two singleton dictionaries, and the goal of the adversary is to determine which. The expression hides passwords computationally if the adversary has only a negligible chance of success.

## 6 Soundness theorems

Our soundness theorems link the symbolic definitions for expression equivalence and secure use of passwords to their computational counterparts. The theorem on expression equivalence can be regarded as an extension of the main theorem of Abadi and Rogaway [1] to the richer language of expressions of this paper.

**Theorem 1 (Soundness for expression equivalence).** *Let $\Pi$ be a secure encryption suite and let $D$ be a dictionary. For any two acyclic expressions $E_0, E_1 \in \mathsf{Exp}$ we have that $E_0 \cong E_1$ implies $[\![E_0]\!]_{\Pi,\mathsf{D}} \approx [\![E_1]\!]_{\Pi,\mathsf{D}}$.*

Our main theorem says that, under certain hypotheses, if the use of passwords is secure symbolically, then it is also secure computationally.

**Theorem 2 (Soundness of password hiding).** *Let $\Pi$ be a secure encryption suite and let $\mathsf{D}$ be a dictionary. For any acyclic expression $E \in \mathsf{Exp}$ if $E$ hides passwords symbolically then $E$ hides passwords in $\mathsf{D}$ computationally.*

A question that we do not investigate in this paper is under what conditions the converses of Theorems 1 and 2 hold. However, it seems quite likely that the techniques and the assumptions for proving completeness of symbolic equivalence for the case of symmetric encryption (e.g. [19]) extend to the setting of this paper.

As an example, we show how to apply our results in the case of the influential Encrypted Key Exchange (EKE) protocol [5]. In the language Exp, the flows of the protocol between parties $A$ and $B$ that share a password $W$ are as follows.

1. $A$ generates an asymmetric key pair $(K_1^e, K_1^d)$ and sends $\{K_1^e\}_W$ to $B$.
2. $B$ decrypts this message using $W$. Then $B$ generates a symmetric key $K_1^s$ and sends $\{\{K_1^s\}_{K_1^e}\}_W$ to $A$.
3. At this point the parties share the key $K_1^s$, and check if the protocol was executed as expected: $A$ generates a symmetric key $K_A^s$ and sends $\{K_A^s\}_{K_1^s}$ to $B$.
4. Upon receiving this message, $B$ obtains $K_A^s$, generates a new symmetric key $K_B^s$, and sends $\{(K_A^s, K_B^s)\}_{K_1^s}$ to $A$. (In the original protocol, $K_A^s$ and $K_B^s$ are random nonces; for simplicity we model these nonces as random symmetric keys.)
5. $A$ decrypts this message and checks that the first component of the resulting pair is $K_A^s$. If so, it obtains $K_B^s$, sends $\{K_B^s\}_{K_1^s}$ to $B$, and terminates successfully.
6. Finally, $B$ decrypts this last message, verifies that it contains the key $K_B^s$ it previously sent to $A$, and if so, it terminates successfully.

A transcript of the execution of the protocol is given by the expression:

$$E = \{K_1^e\}_W, \{\{K_1^s\}_{K_1^e}\}_W, \{K_A^s\}_{K_1^s}, \{(K_A^s, K_B^s)\}_{K_1^s}, \{K_B^s\}_{K_1^s}$$

Since $pattern(E) = \{K_1^e\}_W, \{\Box^a\}_W, \Box^s, \Box^s, \Box^s$ is the instantiation of a pattern in $\mathsf{Pat}[\mathsf{Var}]$ with an appropriate mapping, by definition, $E$ hides the password symbolically. It follows from Theorem 2 that $E$ also hides the password computationally. Informally, this means that for any probabilistic polynomial-time adversary, the probability that the adversary can determine correctly which of two passwords $w_0$ and $w_1$ was used in a given protocol execution is negligible. Once we have Theorem 2, the proof of this fact via the formal definitions is much simpler than a computational proof from scratch.

## 7 Conclusions

In this paper we investigate the use of password-based encryption schemes in protocols from the perspective of a recent line of research aimed at bridging the gap between the symbolic and computational views of cryptography. We give symbolic and computational interpretations to the elements of a language of formal expressions built using symmetric, asymmetric, and password-based encryption. We then prove that symbolic accounts of expression equivalence and password hiding imply strong, computational formulations of the same properties. We base our results on a new computational security definition for password-based encryption, which may be of independent interest.

Off-line guessing attacks, as typically considered in the literature, are inherently passive: an adversary, with some data about a protocol execution, analyzes the data in an attempt to obtain information about the password in use. Our definitions and theorems focus strictly on the data analysis, and do not consider how the data is obtained. Thus, we neither address nor exclude the possibility that the adversary may play a role in protocol executions, perhaps mounting standard active attacks, and obtaining data from interactions with other participants. For protocols that do not rely on passwords, research on the relations between symbolic and computational models has recently dealt with active attacks (e.g., [2, 20]). In further work, it may be worthwhile to integrate the results of that research with the present analysis of password-based encryption.

*Acknowledgements.* We thank Bruno Blanchet, Cédric Fournet, and Phil Rogaway for helpful discussions on password-based protocols, and thank Mathieu Baudet and Nathan Whitehead for suggesting improvements to the presentation of this paper. This research was partly carried out while the second author was at the University of California at Santa Cruz, and it was partly supported by the National Science Foundation under Grants CCR-0204162, CCR-0208800, and ITR-0430594.

## References

1. M. Abadi and P. Rogaway. Reconciling two views of cryptography (The computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, 2002.
2. M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, pages 220–330, 2003.
3. M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *Advances in Cryptology — ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 531–545. Springer-Verlag, 2000.
4. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155. Springer-Verlag, 2000.
5. S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proceedings of the 1992 IEEE Symposium on Security and Privacy*, pages 72–84, 1992.
6. V. Boyko, P. MacKenzie, and S. Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 156–171. Springer-Verlag, 2000.

7. 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.
8. 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)*, pages 62–71, 2003.
9. S. Delaune and F. Jacquemard. A theory of dictionary attacks and its complexity. In *Proc.of the 17th IEEE Computer Security Foundations Workshop (CSFW 2004)*, pages 2–15, 2004.
10. R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. In *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 524–543. Springer-Verlag, 2003.
11. O. Goldreich and Y. Lindell. Session key generation using human passwords only. In *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *LNCS*, pages 403–432. Springer-Verlag, 2001.
12. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.
13. L. Gong. Verifiable-text attacks in cryptographic protocols. In *INFOCOM '90*, pages 686–693, 1990.
14. 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.
15. J. Katz, R. Ostrovsky, and M. Yung. Practical password-authenticated key exchange provably secure under standard assumptions. In *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 475–494. Springer-Verlag, 2001.
16. J. Kohl and C. Neuman. RFC 1510: The Kerberos network authentication service (V5). Web page at ftp://ftp.isi.edu/in-notes/rfc1510.txt, 1993.
17. P. Laud. Symmetric encryption in automatic analyses for confidentiality against active adversaries. In *Proc. of 2004 IEEE Symposium on Security and Privacy*, pages 71–85, 2004.
18. G. Lowe. Analysing protocols subject to guessing attacks. *Journal of Computer Security*, 12(1):83–98, 2004.
19. D. Micciancio and B. Warinschi. Completeness theorems for the Abadi-Rogaway logic of encrypted expressions. *Journal of Computer Security*, 12(1):99–129, 2004.
20. D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In *Theory of Cryptography Conference (TCC 2004)*, volume 2951 of *LNCS*, pages 133–151. Springer-Verlag, 2004.
21. G. Miklau and D. Suciu. Controlling access to published data using cryptography. In *Proceedings of 29th International Conference on Very Large Data Bases – VLDB 2003*, pages 898–909. Morgan Kaufmann Publishers, 2003.
22. J. Mitchell, A. Ramanathan, A. Scedrov, and V. Teague. A probabilistic polynomial-time calculus for analysis of cryptographic protocols. *Electronic Notes in Theoretical Computer Science*, 45, 2001.
23. S. Patel. Number theoretic attacks on secure password schemes. In *Proc. of the IEEE Symposium on Research in Security and Privacy*, pages 236–247, 1997.