Symbolic Probabilistic Analysis of Off-line Guessing

Bruno Conchinha[1], David Basin[1], and Carlos Caleiro[2]

[1] Institute of Information Security, ETH Zurich, Switzerland
[2] SQIG - Instituto de Telecomunicações, Department of Mathematics, IST, TU Lisbon, Portugal,

Abstract. We introduce a probabilistic framework for the automated analysis of security protocols. Our framework provides a general method for expressing properties of cryptographic primitives, modeling an attacker more powerful than conventional Dolev-Yao attackers. It allows modeling equational properties of cryptographic primitives as well as property statements about their weaknesses, e.g. primitives leaking partial information about messages or the use of weak random generation algorithms. These properties can be used to automatically find attacks and estimate their success probability. Existing symbolic methods can neither model such properties nor find such attacks. We show that the probability estimates we obtain are negligibly different from those yielded by a generalized random oracle model based on sampling terms into bitstrings while respecting the stipulated properties of cryptographic primitives.
As case studies, we use a prototype implementation of our framework to model non-trivial properties of RSA encryption and automatically estimate the probability of off-line guessing attacks on the EKE protocol.

Keywords: Probability, Off-line Guessing, Equational Theories, Random Oracle Model

1 Introduction

Cryptographic protocols play an important role in securing distributed computation and it is crucial that they work correctly. Symbolic verification approaches are usually based on the *Dolev-Yao model*: messages are represented by terms in a term algebra, cryptography is assumed to be perfect, and properties of cryptographic operators are formalized equationally [1]. This strong abstraction eases analysis and numerous successful verification tools rely on it [2, 3]. However, it may not accurately represent an attacker's capabilities. As a consequence, broad classes of attacks that rely on weaknesses of cryptographic primitives fall outside the scope of such methods. In contrast, proving security by reasoning directly about bitstrings, as in *computational approaches* [4, 5], yields stronger security guarantees. However, it requires long, error-prone, hand-written proofs to establish the security of given protocols using specific cryptographic primitives.

Much research has been devoted to bridging the gap between these two methods [6]. Below we discuss existing approaches in greater detail.

Related work. There are two main lines of research that aim to bridge the gap between symbolic and computational models: (1) obtaining computational soundness results for symbolic methods, and (2) developing techniques that reason directly with computational models.

The first line of research, developing computational soundness results, was initiated with Abadi and Rogaway's seminal paper [7]. They investigated assumptions under which security against a Dolev-Yao attacker (easier to verify) implies computational security (much stronger). Many such results are now known, e.g. [8–10]. However, such results require strong assumptions on the security of cryptographic primitives. Moreover, messages must be tagged so that their structure is known to any observer, and extending the results to new primitives often involves re-doing most of the work.

The second line of research aims to automate computational security proofs, by formulating security properties in terms of games and obtaining a sequence of security-preserving transformations between such games. Such methods have been implemented by tools like CryptoVerif [11], CertiCrypt [12], and EasyCrypt [13]. When successful, these tools can prove protocols computationally correct and provide upper bounds on the probability of an attack. [14, 15] propose another approach: an automatable, symbolic framework in which it is possible to express security properties of cryptographic primitives and use them to prove computational protocol security.

A limitation of all of the above approaches is that they can only be used to prove security. Failure to obtain a security proof does not imply that an attack exists. Therefore, their usefulness remains limited when cryptographic primitives are too weak to meet the assumptions of their methods.

Our applications in this paper focus on off-line guessing attacks. Given the pervasive use of weak human-picked passwords, off-line guessing attacks are a major concern in security protocol analysis and have been the subject of much research. Symbolic [16, 17] and computational approaches [18] have been used, and computational soundness results [19, 20] relate the two. However, off-line guessing attacks remain a real threat to protocol security. Password-cracking software is freely available on the Internet, and is remarkably successful [21]. Furthermore, such attacks often rely on weaknesses of cryptographic primitives outside the scope of existing automated methods [22, 23].

Contributions. We present a fundamentally new approach to strengthening the security guarantees provided by automated methods. Our approach is in a sense dual to current research that aims to bridge the gap between symbolic and computational models: Rather than assuming strong security properties of cryptographic primitives and using them to prove security, we explicitly describe weaknesses of cryptographic primitives and random number generation algorithms and use them to find attacks.

We propose a probabilistic framework for security protocol analysis in which properties of cryptographic primitives can be specified. Besides equational properties, our framework allows us to express security relevant properties of random number generation algorithms and relations between the input and the output of

cryptographic primitives. For instance, it can model a random number generation algorithm that generates bitstrings representing primes of a certain length, a hash function that leaks partial information about the original message, or a cryptosystem whose valid public keys have some recognizable structure. The specified properties can then be used to find attacks and to estimate their success probability. Such properties cannot be modeled by existing symbolic methods and often lead to attacks on real-world implementations.

We model cryptographic functions using a generalized random oracle model. Given a specification of the cryptographic primitives used and their properties, symbolic terms are sampled to bitstrings in a way that ensures that the specification properties are always satisfied, but otherwise functions behave as random oracles. Under reasonable assumptions on the specification, we can define such generalized random oracles and prove that they yield valid probability measures. Moreover, we show that probabilities in this model can be effectively computed, and we provide a prototype implementation that calculates these probabilities. We believe that this model is interesting in its own right. It is a non-trivial generalization of the standard model of random oracle for hash functions, and it captures the intuitive idea that cryptographic primitives satisfy stated properties, which can be exploited by an attacker, but otherwise behave ideally.

We illustrate the usefulness of our framework by representing the redundancy of RSA keys and using this to model and estimate the success probability of off-line guessing attacks on variants of the EKE protocol [22]. Although these attacks are well-known, their analysis was previously outside the scope of symbolic methods. Potential further applications of our approach include reasoning about differential cryptanalysis or side-channel attacks [24], as well as short-string authentication and distance-bounding protocols.

Outline. In Section 2 we describe our framework's syntax and semantics. In Section 3 we introduce our generalized random oracle model and show that it yields a computable probability measure. In Section 4 we show how our framework can be used to find off-line guessing attacks. In Section 5 we draw conclusions and discuss future work. Our technical report [25] provides full proofs of all results.

2 Definitions

In this section we introduce the syntax and semantics of our framework.

2.1 Setup specification

Term algebra. A *signature* $\Sigma = \biguplus_{n\in\mathbb{N}} \Sigma_n$ is a set of function symbols, where Σ_i contains the symbols of arity i. Given a set G of generators, we define $T_\Sigma(G)$ as the smallest set such that $G \subseteq T_\Sigma(G)$, and if $f \in \Sigma_n$ and $t_1, \ldots, t_n \in T_\Sigma(G)$, then $f(t_1, \ldots, t_n) \in T_\Sigma(G)$. If $c \in \Sigma_0$, we write c instead of $c()$. Unless otherwise stated, we will consider $G = \emptyset$ and write T_Σ instead of $T_\Sigma(\emptyset)$. We define the *head* of a term $t = f(t_1, \ldots, t_n)$ by $\mathit{head}(t) = f$. The set $\mathit{sub}(t)$ of *subterms* of a

term t is defined as usual. The set $psub(t)$ of *proper subterms* of t is $psub(t) = sub(t) \setminus \{t\}$. If $f\colon A \to B$ and $A' \subseteq A$, we write $f[A']$ for the set $\{f(a) \mid a \in A'\}$.

Given a signature Σ, an *equational theory* $\approx$ is a congruence relation on T_Σ. We write $t \approx t'$ instead of $(t, t') \in \approx$. We consider an equational theory $\approx_R$ obtained from a subterm convergent rewriting system R, as in [26].

Property statements. We assume fixed a set $\mathcal{T}$ of *types*. Given a signature Σ, a *property statement* is a tuple $(f, T_1, \ldots, T_n, T)$, written $f[T_1, \ldots, T_n] \subseteq T$, where $f \in \Sigma_n$ and $T_1, \ldots, T_n, T \in \mathcal{T}$. Property statements represent properties of function symbols by expressing relations between their inputs and outputs. If $ps = (f[T_1, \ldots, T_n] \subseteq T)$, we define the *head symbol* of ps by $head(ps) = f$, $dom(ps) = T_1 \times \ldots \times T_n$ and $ran(ps) = T$.

Given a set PS of property statements and $f \in \Sigma$, we denote by PS_f the set of property statements in PS whose head symbol is f. Note that, in general, we may have more than one property statement associated to each function symbol. We write $f[T_1, \ldots, T_n] \subseteq_{PS} T$ instead of $(f[T_1, \ldots, T_n] \subseteq T) \in PS$.

Syntax. The syntax of our setup is defined by a four-tuple $\langle \Sigma, \approx_R, \mathcal{T}, PS \rangle$, where Σ is a signature, $\approx_R$ is an equational theory on T_Σ defined by a convergent rewriting system R, $\mathcal{T}$ is a set of types, and PS is a set of property statements.

We require that Σ_0 is infinite and that $\Sigma \setminus \Sigma_0$ is finite. Symbols in Σ_0 represent either cryptographically relevant constants (e.g., the constant bitstring 0) or random data generated by agents or the attacker.

Interpretation functions. Let $\mathcal{B} = \{0, 1\}$. A *type interpretation function* is a function $[\![\cdot]\!]\colon \mathcal{T} \to \mathcal{P}(\mathcal{B}^*)$ associating each type $T \in \mathcal{T}$ to a finite and non-empty set $[\![T]\!]$. We extend $[\![\cdot]\!]$ to tuples by defining $[\![T_1 \times \ldots \times T_n]\!] = [\![T_1]\!] \times \ldots \times [\![T_n]\!]$.

A *setup specification* is a pair $\mathcal{S} = \langle S, [\![\cdot]\!] \rangle$, where $S = \langle \Sigma, \approx_R, \mathcal{T}, PS \rangle$ defines the setup's syntax as in the above paragraph and $[\![\cdot]\!]$ is a type interpretation function which consistently defines the behavior of all function symbols: that is, $PS_f \neq \emptyset$ for all $f \in \Sigma$ and, whenever $ps_1, ps_2 \in PS_f$, we have $[\![dom(ps_1)]\!] \cap [\![dom(ps_2)]\!] = \emptyset$. For $c \in \Sigma_0$, this implies that there is a single $T \in \mathcal{T}$ such that $c \subseteq_{PS} T$. We denote this unique type T by $type(c)$.

We assume that functions are undefined unless otherwise specified by a property statement: That is, if $f \in \Sigma_n$ and there is no $ps \in PS_f$ such that $(b_1, \ldots, b_n) \in [\![dom(ps)]\!]$, then the function represented by f is undefined on the input $(b_1, \ldots, b_n)$. In light of this, we set the *domain of definability* of f to be $dom_{\mathcal{S}}(f) = \biguplus_{ps \in PS_f} [\![dom(ps)]\!]$. Note that $\emptyset \subset dom_{\mathcal{S}}(f) \subseteq (\mathcal{B}^*)^n$ for all $f \in \Sigma_n$.

Example 1. We specify a simple yet realistic setup that includes: a hash function h that maps any bitstring to a bitstring of length 256; a pairing function $\langle \cdot, \cdot \rangle$ that, given any pair of bitstrings, returns their labeled concatenation; and a symmetric encryption scheme $\{\!|\cdot|\!\}_\cdot$ that uses a block cipher together with some reversible padding technique. The corresponding signature $\Sigma^{\mathcal{DY}}$ is given by $\Sigma^{\mathcal{DY}} = \Sigma_0^{\mathcal{DY}} \cup \Sigma_1^{\mathcal{DY}} \cup \Sigma_2^{\mathcal{DY}}$, where $\Sigma_0^{\mathcal{DY}}$ is a countably infinite set of constant symbols, $\Sigma_1^{\mathcal{DY}} = \{\mathsf{h}, \pi_1, \pi_2\}$ and $\Sigma_2^{\mathcal{DY}} = \left\{ \{\!|\cdot|\!\}_\cdot, \{\!|\cdot|\!\}_\cdot^{-1}, \langle \cdot, \cdot \rangle \right\}$.

Standard equational properties of these primitives are represented by the rewriting system $R_{\mathcal{DY}}$ containing the rules $\pi_1(\langle x, y\rangle) \to x$, $\pi_2(\langle x, y\rangle) \to y$, and $\left\{\left|\{\!|x|\!\}_y\right|\right\}_y^{-1} \to x$. It is simple to check that this rewriting system is convergent.

The types we will consider and their interpretations under $[\![\cdot]\!]$ are as follows. Weak (e.g., human-chosen) passwords are represented by the type pw. We model these passwords as 256-bit bitstrings sampled from a small set: thus, $[\![\mathsf{pw}]\!] \subset \mathcal{B}^{256}$ and $|[\![\mathsf{pw}]\!]| = 2^{24}$. Symmetric keys are represented by the type $\mathsf{sym\_key}$, with $[\![\mathsf{sym\_key}]\!] = \mathcal{B}^{256}$; text represents one block of plaintext, with $[\![\mathsf{text}]\!] = \mathcal{B}^{256}$. Furthermore, for each $n, m \in \mathbb{N}$, we consider the following types: $T_{\mathcal{B}^n}$, with $[\![T_{\mathcal{B}^n}]\!] = \mathcal{B}^n$; $T_{\mathcal{B}^{(n,m)}}$, with $[\![T_{\mathcal{B}^{(n,m)}}]\!] = \mathcal{B}^{(n,m)} = \bigcup_{i=n}^{m} \mathcal{B}^i$; and $T_{\mathcal{B}^{n\#m}}$, with $[\![T_{\mathcal{B}^{n\#m}}]\!] = \mathcal{B}^{n\#m} \subseteq \mathcal{B}^{n+m+\lceil \log(n+m)\rceil}$, representing the set of labeled concatenations of two bitstrings of size n and m.

We define PS as the set that contains all property statements of the form $h[T_{\mathcal{B}^n}] \subseteq T_{\mathcal{B}^{256}}$, $\pi_1[T_{\mathcal{B}^{n\#m}}] \subseteq T_{\mathcal{B}^n}$, $\pi_2[T_{\mathcal{B}^{n\#m}}] \subseteq T_{\mathcal{B}^m}$, $\langle T_{\mathcal{B}^n}, T_{\mathcal{B}^m}\rangle \subseteq T_{\mathcal{B}^{n\#m}}$, $\{\!|T_{\mathcal{B}^{(256n+1,256(n+1)}}|\!\}_{T_{\mathcal{B}^{256}}} \subseteq T_{\mathcal{B}^{256(n+1)}}$ or $\{\!|T_{\mathcal{B}^{256(n+1)}}|\!\}^{-1}_{T_{\mathcal{B}^{256}}} \subseteq T_{\mathcal{B}^{(256n+1,256(n+1))}}$, for $n, m \in \mathbb{N}$. Note that all functions are modeled as undefined on all arguments that fall outside the domains of these property statements. For example, the encryption of any term is undefined unless the key is a 256-bit bitstring.

Example 2. We use our framework to formalize RSA encryption, taking into account properties of the key generation algorithm. An RSA public key is a pair (n, e), where $n = p \cdot q$ is the modulus (with p and q being large primes, typically of around 512 bits), and the exponent e is coprime to $\varphi(n) = (p-1)(q-1)$. The private key d is the multiplicative inverse of e modulo $\varphi(n)$.

We extend the setup specification of Example 1. We add to the signature the following five primitives: the unary functions mod, expn, and inv, representing the extraction of the modulus, the exponent, and the exponent's multiplicative inverse, respectively, from an RSA public-private key pair; a binary function $\{\cdot\}_\cdot^{-1}$, representing the RSA decryption function; and a ternary function $\{\cdot\}_{\cdot,\cdot}$, representing RSA encryption. The only rewriting rule that we must add to model RSA encryption is $\left\{\{m\}_{\mathsf{mod}(k),\mathsf{expn}(k)}\right\}^{-1}_{\mathsf{inv}(k)} \to m$, where m and k are variables.

The additional types that we will use to model properties of these functions and their interpretations are as follows: random represents the random values used to generate an RSA public-private key pair, including two 512-bit prime numbers and the 1024-bit exponent, with $[\![\mathsf{random}]\!] \subseteq \mathcal{B}^{2048}$; $\mathsf{prodprime}$ represents the product of two 512-bit prime numbers, so that $[\![\mathsf{prodprime}]\!] \subseteq \mathcal{B}^{1024}$, and, by the prime number theorem, $|[\![\mathsf{prodprime}]\!]| \approx (2^{512}/\log(2^{512}))^2 \approx 2^{1009}$; odd represents 1024-bit odd numbers, with $[\![\mathsf{odd}]\!] \subseteq \mathcal{B}^{1024}$ and $|[\![\mathsf{odd}]\!]| = 2^{1023}$.

The additional property statements we include are the following: $\mathsf{mod}[\mathsf{random}] \subseteq \mathsf{prodprime}$, because the modulo of an RSA public key is the product of two primes; $\mathsf{expn}[\mathsf{random}] \subseteq \mathsf{odd}$, because the exponent of an RSA public key is always odd; $\mathsf{inv}[\mathsf{random}] \subseteq T_{\mathcal{B}^{1024}}$, because an RSA private key is a 1024-bit bitstring (note that we do not allow extracting modulus, exponents, or inverses from anything other than a valid value for generating an RSA key pair);

$\{T_{\mathcal{B}^{1024}}\}_{\mathsf{prodprime,odd}} \subseteq T_{\mathcal{B}^{1024}}$; and $\{T_{\mathcal{B}^{1024}}\}^{-1}_{T_{\mathcal{B}^{1024}}} \subseteq T_{\mathcal{B}^{1024}}$. The last two properties state that encrypting any 1024-bit plaintext with a valid RSA public key yields a 1024-bit bitstring, and that RSA decryption takes a ciphertext and a private key which are both 1024-bit bitstrings and outputs a 1024-bit plaintext. Note that encryption is undefined if the plaintext is not a 1024-bit bitstring, the modulus is not the product of two primes, or the exponent is even.

One limitation of our method is that, although it is simple to express relations between the input and output of a cryptographic primitive, more complex relations between terms are harder to model. For example, modeling the fact that the $\varphi(n)$ and e are coprime would require modeling the public key as a single term. An attacker could then extract the modulus and the exponent from such a key, and it can build such a key from a modulus and an exponent. The simpler model we present here illustrates the expressiveness of our framework and is sufficient to model the attacks in our case studies.

2.2 Semantics

Let us fix a setup specification $\mathcal{S} = \langle\langle \Sigma, \approx_R, \mathcal{T}, PS \rangle, [\![\cdot]\!]\rangle$.

Term assignments. Let $\mathcal{B}^*_\perp = \mathcal{B}^* \cup \{\perp\}$. A term assignment is a function $\omega : T_\Sigma \to \mathcal{B}^*_\perp$ associating a bitstring to each symbolic term. Let $\mathbf{\Omega}$ be the set of all term assignments. We say that $\omega \in \mathbf{\Omega}$ *satisfies* $\approx_R$, and write $\omega \models \approx_R$, if, whenever $t \approx_R t'$, either $\omega(t) = \omega(t')$, or $\omega(t) = \perp$, or $\omega(t') = \perp$. We say that ω satisfies a property statement ps (under $[\![\cdot]\!]$), and write $\omega \models_{[\![\cdot]\!]} ps$ if, whenever $(\omega(t_1), \ldots, \omega(t_n)) \in [\![dom(ps)]\!]$, then $\omega(f(t_1, \ldots, t_n)) \in [\![ran(ps)]\!]$, and whenever $(\omega(t_1), \ldots, \omega(t_n)) \notin [\![dom(ps)]\!]$ for all $ps \in PS_f$, then $\omega(f(t_1, \ldots, t_n)) = \perp$. We say that ω satisfies PS (under $[\![\cdot]\!]$), and write $\omega \models_{[\![\cdot]\!]} PS$, if $\omega \models_{[\![\cdot]\!]} ps$ for all $ps \in PS$. We say that ω satisfies $\mathcal{S}$, and write $\omega \models \mathcal{S}$, when $\omega \models \approx_R$ and $\omega \models_{[\![\cdot]\!]} PS$. We denote by $\Omega_\mathcal{S}$ the set of all $\omega \in \mathbf{\Omega}$ which satisfy $\mathcal{S}$.

Example 3. Functions ω that satisfy our equational theory may be such that $\omega(t) = \perp$ and $\omega(t') \neq \perp$ for terms t and t' such that $t \approx_R t'$. To see why this is allowed, recall from Example 1 that $\{\!|\cdot|\!\}^{-1}_\cdot$ represents a symmetric encryption algorithm in which valid keys always have 256 bits. Let $t, k \in \Sigma_0$, with $type(t) = \mathsf{text}$, and $t' = \{\!|\{\!|t|\!\}_k|\!\}^{-1}_k$. We have $t \approx_R t'$. If ω represents a possible real-world assignment (of terms to bitstrings), we have $\omega(t) \neq \perp$ (since t represents a bitstring freshly sampled from $\mathcal{B}^{256}$). Moreover, if $\omega(k)$ is not a 256-bit bitstring, then $\omega(t') = \perp$ since our encryption and decryption functions are only defined for 256-bit keys. Therefore, $\omega(\{\!|\{\!|t|\!\}_k|\!\}^{-1}_k) = \perp$.

Probabilistic models. Since valid, real-world protocol execution traces are finite, we are interested in events that depend on finitely many terms. For each finite set of terms $K \subseteq T_\Sigma$, let Λ_K be the set of functions $\lambda : K \to \mathcal{P}(\mathcal{B}^*_\perp)$ and, for each $\lambda \in \Lambda_K$, let Ω_λ be the set of all $\omega \in \mathbf{\Omega}$ such that $\omega(t) \in \lambda(t)$ for all $t \in K$. Let $\Lambda = \bigcup_{K \in \mathcal{P}_{fin}(T_\Sigma)} \Lambda_K$ and $\Omega_\Lambda = \{\Omega_\lambda \mid \lambda \in \Lambda\}$, where $\mathcal{P}_{fin}(X)$ is the set of *finite* subsets of X. Note that Ω_Λ is the set of subsets of $\mathbf{\Omega}$ whose specification

depends on only the instantiation of finitely many terms. Thus, we want our probability measure to be defined in the σ-algebra generated by Ω_A. Let $\mathcal{F}$ be this σ-algebra; we say that $\mathcal{F}$ is the σ-algebra of *finitely generated events*.

We consider probability spaces $(\mathbf{\Omega}, \mathcal{F}, \mu)$, where $\mathbf{\Omega}$ and $\mathcal{F}$ are as defined above and $\mu\colon \mathcal{F} \to [0,1]$ is a probability measure. Note that $\mathbf{\Omega}$ and $\mathcal{F}$ are fixed for a given $\mathcal{S}$; it is μ that we are interested in studying. If $t \in T_\Sigma$, we write $\widehat{t}\colon \mathbf{\Omega} \to \mathcal{B}_\perp^*$ to denote the random variable on $\mathbf{\Omega}$ defined by $\widehat{t}(\omega) = \omega(t)$. We adopt standard (abuses of) notation from probability theory. If $C(b_1, \ldots, b_n)$ is a condition whose satisfaction depends on the bitstring values $b_1, \ldots, b_n$, we write $P_\mu[C(\widehat{t_1}, \ldots, \widehat{t_n})]$ for $\mu(\{\omega \in \mathbf{\Omega} \mid C(\widehat{t_1}(\omega), \ldots, \widehat{t_n}(\omega))\})$, provided that $\{\omega \in \mathbf{\Omega} \mid C(\widehat{t_1}(\omega), \ldots, \widehat{t_n}(\omega))\} \in \mathcal{F}$. If $\Omega \in \mathcal{F}$, we write $P_\mu[\Omega]$ instead of $\mu(\Omega)$. We say μ *satisfies* the equational theory $\approx_R$ if $\mu(\{\omega \mid \omega \models \approx_R\}) = 1$, and we write $\mu \models \approx_R$ to denote this fact. Analogously, we define the satisfaction of PS (under $[\![\cdot]\!]$) by μ, $\mu \models_{[\![\cdot]\!]} PS$, by $\mu(\{\omega \mid \omega \models_{[\![\cdot]\!]} PS\}) = 1$. We say that μ *satisfies*, or is a *model* of, the setup specification $\mathcal{S}$, written $\mu \models \mathcal{S}$, if $\mu \models \approx_R$ and $\mu \models_{[\![\cdot]\!]} PS$. Note that μ is a model of $\mathcal{S}$ if and only if $\mu(\Omega_{\mathcal{S}}) = 1$.

3 A Generalized Random Oracle Model

In this section we propose an algorithm for sampling the random variables associated with symbolic terms. Our algorithm interprets functions as random oracles subject to satisfying our setup specification $\mathcal{S} = \langle\langle \Sigma, \approx_R, \mathcal{T}, PS \rangle, [\![\cdot]\!]\rangle$.

3.1 Tentative term sampling in the ROM

Term sampling. Suppose that $K \subset T_\Sigma$ is a finite set of terms and P is a partition of K. We define $\approx_P$ to be the smallest congruence relation on T_Σ such that $\approx_R \subseteq \approx_P$ and $t \approx_P t'$ whenever there is $p \in P$ such that $t, t' \in p$. Note that $\approx_P$ may be coarser than both $K/\approx_R$ and P: For example, if there are $a, b \in \Sigma_0$ and $p \in P$ such that $a, b \in p$, then $\{\!|\{\!|M|\!\}_a|\!\}_b^{-1} \approx_P M$. However, $\{\!|\{\!|M|\!\}_a|\!\}_b^{-1} \not\approx_R M$ and there is not necessarily a $p \in P$ such that $M, \{\!|\{\!|M|\!\}_a|\!\}_b^{-1} \in p$.

The sampling algorithm below builds a function ψ_{ROM} mapping a finite set of terms to $\mathcal{B}_\perp^*$. We denote by $P(\psi_{ROM})$ the partition of $dom(\psi_{ROM})$ given by $P(\psi_{ROM}) = \{\psi_{ROM}^{-1}(b) \mid b \in ran(\psi_{ROM})\}$. The algorithm is probabilistic: at various steps, it samples a random bitstring from a finite subset of $\mathcal{B}_\perp^*$. We assume that this sampling is always done with uniform probability distribution. We also assume fixed some total order $\prec$ on the set of terms such that, if $t \in psub(t')$, then $t \prec t'$. We say that such an order is *subterm-compatible*.

Algorithm 1 (Tentative Term Sampling Algorithm)
Input: *a finite set of terms* $K \subseteq T_\Sigma$.
Output: *a function* $\psi_{ROM}\colon sub[K] \to \mathcal{B}_\perp^*$.

1: $\psi_{ROM} \leftarrow \emptyset$
2: *let* $t_1, \ldots, t_k$ *be such that* $t_1 \prec \ldots \prec t_k$ *and* $sub[K] = \{t_1, \ldots, t_k\}$
3: **for** i **from** 1 **to** k

4: $\quad$ *let* $t_i = f(t'_1, \ldots, t'_n)$
5: $\quad$ **if** $(\psi_{ROM}(t'_1), \ldots \psi_{ROM}(t'_n)) \notin dom_{\mathcal{S}}(f)$
6: $\quad\quad$ $\psi_{ROM}(t_i) \leftarrow \bot$
7: $\quad\quad$ **continue**
8: $\quad$ *let ps be the unique* $ps \in PS_f$ *s.t.* $(\psi_{ROM}(t'_1), \ldots \psi_{ROM}(t'_n)) \in [\![dom(ps)]\!]$
9: $\quad$ **if** $\exists t' \in dom(\psi_{ROM}).\ t_i \approx_{P(\psi_{ROM})} t'$ ***and*** $\psi_{ROM}(t') \neq \bot$
10: $\quad\quad$ $\psi_{ROM}(t_i) \leftarrow \psi_{ROM}(t')$
11: $\quad\quad$ **continue**
12: $\quad$ *randomly sample b from* $[\![ran(ps)]\!]$
13: $\quad$ $\psi_{ROM}(t_i) \leftarrow b$
14: **return** ψ_{ROM}

Algorithm 1 samples terms in order (lines 2–3), by interpreting each function symbol as a random oracle with uniform probability distribution (lines 12–13), and respecting the equational theory in case an equal term has already been sampled (lines 9–10), as long as its argument values (previously sampled) form a tuple in its domain of definability (lines 5–6).

We remark that this procedure is only used to define our probability distribution μ: in general, it may not be feasible to decide membership of the sets $[\![dom(ps)]\!]$ or to sample from $[\![ran(ps)]\!]$. In [25] we describe our algorithm for computing μ.

Problems with the tentative term sampling algorithm. We show that Algorithm 1 does not necessarily yield a probability measure over $\mathcal{F}$ as desired.

Given a finite set $K \subseteq T_\Sigma$ and a subterm-compatible order $\prec$, Algorithm 1 is a probabilistic algorithm, and thus outputs functions $\psi\colon sub[K] \to \mathcal{B}^*_\bot$ with some probability distribution. We would therefore like to define a model μ of $\mathcal{S}$ by defining $\mu(\Omega_\lambda)$ for each generator Ω_λ of $\mathcal{F}$ as the probability that executing Algorithm 1 on input $dom(\lambda)$ yields as output a function ψ_{ROM} such that, for each $t \in dom(\lambda)$, $\psi_{ROM}(t) \in \lambda(t)$.

Unfortunately, the next example shows that this is not well-defined in general. Concretely, we show that there are terms t and t' such that, letting $\lambda_b = \{t \mapsto b, a \mapsto b\}$ for each $b \in \mathcal{B}^*_\bot$, the probability of the set $\bigcup_{b \in \mathcal{B}^*_\bot} \Omega_{\lambda_b}$ depends on the input set K and the order relation $\prec$ considered.

Example 4. Suppose that $a, b, k \in \Sigma_0$ are such that $type(a) = T_{\mathcal{B}^{1024}}$, $type(b) = T_{\mathcal{B}^{1024}}$ and $type(k) = \mathsf{random}$. Consider executing Algorithm 1 on the set $\{t\}$, with $t = \left\{\{a\}_{\mathsf{mod}(k),\mathsf{expn}(k)}\right\}_b^{-1}$. Algorithm 1 outputs a function $\psi\colon sub(t) \to \mathcal{B}^*_\bot$. Let us consider the probability that $\psi(t) = \psi(a)$. It is simple to check that both $\psi(t)$ and $\psi(a)$ are sampled by Algorithm 1 with uniform probability distribution from $\mathcal{B}^{1024}$. Therefore, the probability that $\psi(t) = \psi(a)$ is 2^{-1024}.

Now, consider executing Algorithm 1 on the set $\{t, \mathsf{inv}(k)\}$. If $t \prec \mathsf{inv}(k)$, then the execution of Algorithm 1 will be exactly the same until $\psi(s)$ is sampled for all terms $s \in sub(t)$, and $\psi(\mathsf{inv}(k))$ is only sampled afterwards. Therefore, $\psi(s)$ is sampled according to the same probability distribution for all $s \in sub(t)$, and

the probability that $\psi(t) = \psi(a)$ is still 2^{-1024}. However, if $\mathsf{inv}(k) \prec b$, we have a probability of 2^{-1024} that $\psi(b) = \psi(\mathsf{inv}(k))$. If $\psi(b) = \psi(\mathsf{inv}(k))$, then we have $\psi(t) = \psi(a)$ with probability 1. Otherwise, $\psi(t)$ and $\psi(a)$ will still be sampled from $\mathcal{B}^{1024}$ with uniform probability distribution, and the probability that they are sampled to the same value is again 2^{-1024}. In this case, we conclude that $P[\psi(a) = \psi(t)] = 2^{-1024} \cdot (2 - 2^{-1024}) \neq 2^{-1024}$. Thus, the probability that $\psi(t) = \psi(a)$ depends on both the input set K and the order $\prec$.

Despite the example above, the following result shows that, given a fixed finite set of terms K and a subterm-compatible order $\prec$, Algorithm 1 does yield a probability distribution on the σ-algebra $\mathcal{F}_K$ generated by the set $\{\Omega_\lambda \mid \lambda \in \Lambda_K\}$. We remark that $\mathcal{F}_K$ is the set of σ-algebra of events that depend only on the instantiation of terms in the set K.

Theorem 2. *There is a unique probability distribution $\mu^{K,\prec}\colon \mathcal{F}_{sub[K]} \to [0,1]$ such that, for each $\lambda \in \Lambda_K$, $\mu^{K,\prec}(\Omega_\lambda)$ is the probability that executing Algorithm 1 on input K and using the order $\prec$ yields a function ψ_{ROM} such that, for each $t \in K$, $\psi_{ROM}(t) \in \lambda(t)$.*

3.2 Revised term sampling in the ROM

To avoid problems like the one illustrated by Example 4 we need two additional hypotheses on the setup specification $\mathcal{S}$. We will explicitly distinguish a set of weak function symbols and consider a revised algorithm that uses this distinction. This revised algorithm is equivalent to Algorithm 1 when all functions are treated as weak. We show that, under these hypotheses, we can define a probability measure from this new sampling algorithm, while also simplifying the calculation of probabilities.

Weak terms. We assume fixed a set $\Sigma^W \subseteq \Sigma$ of *weak function symbols*. We say that a term $t \in T_\Sigma$ is *weak* if $head(t) \in \Sigma^W$, and denote by T^W the set of weak terms. Intuitively, weak function symbols are those that represent functions whose outputs are sampled from "small" sets, and a probabilistic model must therefore take into account the possibility of collisions between them. By contrast, non-weak function symbols are those that represent functions whose outputs are sampled from large enough sets, so that ignoring the possibility of collisions changes our probability estimates only negligibly. Theorem 4, stated below, formalizes this idea.

Example 5. In our running example, we consider the set of weak function symbols $\Sigma^W = \{\mathsf{h}\} \cup \{a \in \Sigma_0 \mid a \subseteq_{PS} \mathsf{pw}\}$. That is, a term is weak if it is a hash or if it is derived from a humanly-chosen password. Note that the probability of a collision in a hash function is in fact rather low, and indeed the security of many protocols relies on hash functions being collision-resistant. However, modeling hash functions as weak increases the accuracy of our model while still allowing us to define a consistent probability distribution.

Term sampling revisited. If K and K' are sets of terms and P is a partition of K, we let $P|_{K'} = \{p \cap K' \mid p \in P\}$. Note that $P|_{K'}$ is a partition of $K \cap K'$. We denote by $W(\psi_{ROM})$ the partition $P(\psi_{ROM})|_{T^W}$.

Our revised term sampling algorithm, targeted at solving the anomaly described in Example 4, is the same as Algorithm 1 with the exception that we replace the condition $t_i \approx_{P(\psi_{ROM})} t'$ by $t_i \approx_{W(\psi_{ROM})} t'$ in line 9. Note that this revised sampling algorithm does not necessarily respect congruences, i.e., we may have $\psi_{ROM}(t) = \psi_{ROM}(t')$ and $\psi_{ROM}(f(t)) \neq \psi_{ROM}(f(t'))$. However, this only happens if either t or t' is not weak, in which case the collision $\psi_{ROM}(t) = \psi_{ROM}(t')$ only occurs with negligible probability.

This revised algorithm yields a probability distribution on $\mathcal{F}$ provided that the setup specification $\mathcal{S}$ satisfies two reasonable conditions, described below.

Disjointness. The first condition we require on the specification $\mathcal{S}$ is that weak function symbols do not occur in the rewriting system R.

Intuitively, this disjointness condition implies that the equality of terms depends only on the equalities between their weak subterms. Thus, sampling terms in a different order does not affect any equalities because terms are sampled only after all their subterms are sampled. This condition excludes cases like that described in Example 4: because $\mathsf{inv} \notin \Sigma^W$, even if $\psi_{ROM}(b) = \psi_{ROM}(\mathsf{inv}(k))$, we never have $\left\{\{a\}_{\mathsf{mod}(k),\mathsf{expn}(k)}\right\}_b^{-1} \approx_{W(\psi_{ROM})} a$. The key idea is that equalities between non-weak terms may be disregarded, as they occur only with negligible probability. Ignoring equalities between non-weak terms, besides allowing us to consistently define a probability measure, also simplifies the calculation of probabilities. In [25] we present a simple algorithm for deciding $\approx_P$ (that is, given terms t and t', decide whether $t \approx_P t'$), and thus to perform the test in Line 9 of Algorithms 1 and its revised version.

Compatibility. The second condition we require on our setup is *compatibility*. Let K be a finite set of terms and P be a partition of K. Recall the definition of $\approx_P$ given in Section 3. We say that P is $\approx_R$*-closed* if, for all $t, t' \in K$, whenever $t \approx_P t'$ there is $p \in P$ such that $t, t' \in p$; equivalently, P is $\approx_R$-closed if $\approx_P|_{K\times K} = \{(t, t') \mid$ there exists $p \in P$ such that $t, t' \in p\}$. We are interested in partitions of weak terms. Thus, given a finite set K, we denote by $\mathcal{P}_R^W(K)$ the set of $\approx_R$-closed partitions of $sub[K] \cap T^W$.

A *selection function for* K is a function $\iota\colon sub[K] \to PS \cup \{\bot\}$ such that, for each $t \in sub[K]$, either $\iota(t) = \bot$ or $head(\iota(t)) = head(t)$. Given $\omega \in \mathbf{\Omega}$, we say that ω *satisfies* ι if, for all $t = f(t_1, \ldots, t_n) \in sub[K]$, either $(\omega(t_1), \ldots, \omega(t_n)) \in [\![dom(\iota(t))]\!]$ and $\omega(t) \in [\![ran(\iota(t))]\!]$, or $(\omega(t_1), \ldots, \omega(t_n)) \notin dom_{\mathcal{S}}(f)$ and $\iota(t) = \omega(t) = \bot$. We denote by $I(K)$ the set of selection functions for K, and by $I_{\mathcal{S}}(K) \subseteq I(K)$ the set of selection functions ι for K such that there is $\omega \in \mathbf{\Omega}$ that satisfies ι. In [25] we show that, given a finite set of terms K, $I_{\mathcal{S}}(K)$ is a finite and computable set.

If K is a finite set of terms, a selection function for K determines which property statement applies to each term in $sub[K]$: Indeed, if $\omega \in \mathbf{\Omega}$ satisfies

PS, there exists exactly one selection function $\iota \in I(K)$ satisfied by ω, which associates each term $f(t_1, \ldots, t_n)$ to the unique property statement $ps \in PS_f$ such that $(\omega(t_1), \ldots, \omega(t_n)) \in [\![dom(ps)]\!]$, or $\bot$ if no such ps exists.

The compatibility condition is that, if K is a finite set of terms, $t \in sub[K]$, $P \in \mathcal{P}_R^W(K)$, $\iota \in I_{\mathcal{S}}(K)$, and $\iota(t) \neq \bot$, then there is $t' \in sub(t)$ such that $t \approx_{P|_{psub(t)}} t'$ and, whenever $t'' \in sub[K]$ and $t \approx_{P|_{psub(t)}} t''$, either $\iota(t'') = \bot$ or $[\![ran(\iota(t'))]\!] \subseteq [\![ran(\iota(t''))]\!]$. Intuitively, this condition requires the equational theory $\approx_R$ and the property statements in *PS* to be compatible. It is a basic requirement that should be satisfied by any meaningful setup specification. The following example illustrates this.

Example 6 (Incompatibility between $\approx_R$ and PS). Consider a rewriting system R containing the symmetric decryption rewrite rule $\left\{\left|\{\!|x|\!\}_y\right|\right\}_y^{-1} \to x$ and the property statements $\{\!|T_{\mathcal{B}^{256}}|\!\}_{T_{\mathcal{B}^{256}}}^{-1} \subseteq T_{\mathcal{B}^{128}}$, $\{\!|T_{\mathcal{B}^{256}}|\!\}_{T_{\mathcal{B}^{256}}} \subseteq T_{\mathcal{B}^{256}}$. Let $t' = \{\!|\{\!|t|\!\}_k|\!\}_k^{-1}$, where $t, k \in \Sigma_0$ and $type(t) = type(k) = T_{\mathcal{B}^{256}}$. In this case, we have $\iota(t) = T_{\mathcal{B}^{256}}$ and $\iota(t') = T_{\mathcal{B}^{128}}$ for all selection functions $\iota \in I_{\mathcal{S}}(\{t, t'\})$. We have $t \approx_R t'$, $[\![ran(\iota(t))]\!] = T_{\mathcal{B}^{256}}$, and $[\![ran(\iota(t'))]\!] = T_{\mathcal{B}^{128}}$. Because $\mathcal{B}^{128} \cap \mathcal{B}^{256} = \emptyset$, it follows that there is no $\omega \in \mathbf{\Omega}$ that satisfies $\approx_R$ and *PS*. Note that, having $\{\!|T_{\mathcal{B}^{256}}|\!\}_{T_{\mathcal{B}^{256}}}^{-1} \subseteq T_{\mathcal{B}^{256}}$ instead of $\{\!|T_{\mathcal{B}^{256}}|\!\}_{T_{\mathcal{B}^{256}}}^{-1} \subseteq T_{\mathcal{B}^{128}}$, we could have $type(t) = B$ for any non-empty set $B \subseteq \mathcal{B}^{256}$ without violating our compatibility condition.

Example 7. With the choice of Σ^W given in Example 5, our running example (from Examples 1—3) satisfies the disjointness and compatibility conditions.

Probability measure. Under the disjointness and compatibility conditions, the revised sampling algorithm yields a probability measure μ_{ROM}. For each total subterm-compatible order $\prec$ and each $\lambda \in \Lambda$, let $\mu^{\prec}(\lambda)$ be the probability that executing the revised version of Algorithm 1 on input $dom(\lambda)$ using the order $\prec$ yields a function $\psi_{ROM}: sub[K] \to \mathcal{B}_\bot^*$ such that $\psi_{ROM}(t) \in \lambda(t)$ for all $t \in K$.

Theorem 3. *Suppose that the disjointness and compatibility conditions are satisfied by $\mathcal{S}$ and Σ^W, and let $\prec$ and $\prec'$ be two subterm-compatible orders. If $\lambda, \lambda' \in \Lambda$ are such that $\Omega_\lambda = \Omega_{\lambda'}$, we have $\mu^{\prec}(\lambda) = \mu^{\prec'}(\lambda')$. There exists a unique extension μ_{ROM} of $\mu^{\prec}$ to $\mathcal{F}$ that is a probability measure, and $\mu_{ROM}(\Omega_{\mathcal{S}}) = 1$.*

Theorem 3 implies that μ_{ROM} is well-defined, as it does not depend on the choice of the order $\prec$, and that it is a model of $\mathcal{S}$.

3.3 Comparing probability measures

We describe the relationship between the probability measures $\mu^{K,\prec}$ described in Theorem 2 and the probability measure μ_{ROM} described in Theorem 3.

For each $f \in \Sigma$, let $L_f = \min_{ps \in PS_f} |[\![ran(ps)]\!]|$ and $L = \min_{f \in \Sigma \setminus \Sigma^W} L_f$. Note that, if non-weak terms are always sampled from "large" sets of bitstrings whenever they are defined, then L is large as well. Intuitively, Theorem 4 shows that, in this case, $\mu^{K,\prec}$ and μ_{ROM} coincide except on a set whose probability is "small". More precisely, fixed K, the probability of this set is $\mathcal{O}(1/L)$.

Theorem 4. *For any finite set of terms K, there exists a set $\Omega(K)$ such that, for any subterm-compatible order $\prec$:*

(1) for any $\lambda \in \Lambda_K$, $\mu^{K,\prec}(\Omega_\lambda \cap \Omega(K)) = \mu_{ROM}(\Omega_\lambda \cap \Omega(K))$;
(2) there exists a polynomial function p such that

$$\mu^{K,\prec}(\mathbf{\Omega} \setminus \Omega(K)) = \mu_{ROM}(\mathbf{\Omega} \setminus \Omega(K)) \ \leq |sub[K]|^2 \cdot |I_{\mathcal{S}}(K)| \cdot (1/L).$$

Note that the statement of Theorem 4 is stronger than merely bounding the difference in the probability of sets in Ω_Λ. For example, Theorem 4 implies that the probability of two terms being sampled to the same bitstring as measured by $\mu^{K,\prec}$ and μ_{ROM} also differs by at most $|sub[K]|^2 \cdot |I_{\mathcal{S}}(K)| \cdot (1/L)$.

Asymptotic interpretation. Suppose that, for each $\eta \in \mathbb{N}$, $[\![\cdot]\!]_\eta$ is a type interpretation function and $\mathcal{S}_\eta = \langle\langle \Sigma, \approx_R, \mathcal{T}, PS \rangle, [\![\cdot]\!]_\eta\rangle$ is a setup specification which, together with a set Σ^W of weak function symbols, satisfies the disjointness and compatibility conditions. Assume further that $1/L_\eta$ is negligible as a function of η, where $L_{f,\eta} = \min_{ps \in PS_f} [\![ran(ps)]\!]_\eta$ and $L_\eta = \min_{f \in \Sigma \setminus \Sigma^W} L_{f,\eta}$ for each $\eta \in \mathbb{N}$. Note that this condition is equivalent to requiring, for each function symbol $f \in \Sigma \setminus \Sigma^W$ and each $ps \in PS_f$, that $1/\left|[\![ran(ps)]\!]_\eta\right|$ is negligible as a function of η. Intuitively, this condition requires that non-weak terms, when defined, are always mapped to bitstrings sampled from large enough sets.

Let $\mu_\eta^{K,\prec}$ (respectively, $\mu_{ROM,\eta}$) be the probability measure given by Theorem 2 (respectively, Theorem 3) when Algorithm 1 (respectively, the revised version of algorithm 1) is executed using the interpretation function $[\![\cdot]\!]_\eta$. Then, the following is a corollary of Theorem 4.

Corollary 1. *Let K be a finite set of terms, and suppose that $\left|I_{\mathcal{S}_\eta}(K)\right|$ grows polynomially as a function of η. For any finite set of terms K, there exists a set $\Omega(K)$ such that, for any subterm-compatible order $\prec$:*

(1) for any $\lambda \in \Lambda_K$, $\mu_\eta^{K,\prec}(\Omega_\lambda \cap \Omega(K)) = \mu_{ROM,\eta}(\Omega_\lambda \cap \Omega(K))$;
(2) $\mu_\eta^{K,\prec}(\mathbf{\Omega} \setminus \Omega(K)) = \mu_{ROM,\eta}(\mathbf{\Omega} \setminus \Omega(K))$, and both quantities are negligible as functions of η.

Comparison with the random oracle and ideal cipher models. Algorithm 1 exactly matches the random oracle model for hash functions. Its only difference with respect to the ideal-cipher model for symmetric encryption is that two different bitstrings may be encrypted to the same ciphertext under the same key. However, if the range of the encryption function is large enough (i.e., larger than any polynomial function of the security parameter), then the probability of such a collision is negligible for any (finite) input set. In light of Corollary 1, we thus conclude that the probability measure μ_{ROM} differs only negligibly from the probabilities yielded by the random oracle and the ideal cipher models.

3.4 Computing probabilities

In [25] we present an equivalent, algebraic definition of the probability measure μ_{ROM} which reduces the problem of computing probabilities of the form $P_{\mu_{ROM}}[t_1 \in B_1, \ldots, t_n \in B_n, t'_1 = t''_1, \ldots, t'_{n'} = t''_{n''}]$ (with $B_1, \ldots, B_n \subseteq \mathcal{B}^*_\perp$) to computing the sizes of intersections of sets in $\{B_1, \ldots, B_n\} \cup [\![\mathcal{T}]\!]$. A full specification of the interpretations of types is not necessary.

Our prototype implementation computes probabilities of this form for the cryptographic primitives and respective properties considered in our running example. The user may, however, need to specify the sizes of intersections of the sets of bitstrings $B_1, \ldots, B_n$ with the specified property types.

Let $T = \{t_1, \ldots, t_n, t'_1, t''_1, \ldots, t'_{n'}, t'_{n''}\}$. Since we must consider $\approx_R$-closed partitions of $T^W_\Sigma \cap sub[T]$, the complexity of the computation is exponential in $\left|T^W_\Sigma \cap sub[T]\right|$. However, for the specification considered in our running example, if T contains no subterms of the form $\pi_i(t)$ for $i \in \{1, 2\}$ and t such that $head(t) \neq \langle \cdot, \cdot \rangle$, the complexity is linear in the number of non-weak subterms of T.

4 Off-line guessing

Let s be a term representing a bitstring in $B \subset \mathcal{B}^*$ that is intended to be secret. If an attacker can feasibly enumerate all bitstrings in B, he may try to rule out the possibility that s represents each such bitstring. The attacker's ultimate aim is to exclude all but one bitstring in B and thereby learn the secret s even if it may not be directly deduced by constructing terms and reasoning equationally. When the attacker does not need to interact with other agents to verify his guess, this is called an *off-line guessing attack*. In this section we describe how properties of cryptographic primitives described by $\mathcal{S}$ can be used to find and estimate the success probability of non-trivial off-line guessing attacks.

4.1 Attacker model

We will assume fixed an infinite set $\mathcal{N} \subseteq \Sigma_0$ such that $\Sigma_0 \setminus \mathcal{N}$ is finite. Symbols in $\mathcal{N}$ represent random data generated by the agents, whereas symbols in $\Sigma_0 \setminus \mathcal{N}$ represent cryptographically relevant constants (such as the bitstring 0). We also assume fixed a countably infinite set $\mathcal{V}$ of *variables*, disjoint from Σ.

We represent an attacker's knowledge by a frame [27], i.e., a pair $(\tilde{n}, \sigma)$, written $\nu\tilde{n}.\sigma$, where $\tilde{n} \subseteq \mathcal{N}$ is a finite set of names and $\sigma\colon \mathcal{V} \nrightarrow T_\Sigma$ is a substitution. Given a frame $\phi = \nu\tilde{n}.\sigma$, we define $T_\phi = T_{\Sigma \setminus \tilde{n}}(dom(\sigma))$. We say that terms in T_ϕ are *ϕ-recipes* as they represent the ways in which an attacker can build terms.

Suppose that an attacker whose knowledge is represented by a frame $\phi = \nu\tilde{n}.\sigma$ attempts an off-line guessing attack of a secret s. We require that the set of bitstrings tried by the attacker is $[\![type(w)]\!]$ for some $w \in \mathcal{N}$ that does not occur in either $\tilde{n}$ or σ, and we model the attacker's guess by w. Letting $x \notin dom(\sigma)$ be a fresh variable, we consider the frames $\phi_s = \nu\tilde{n}_w.\sigma_s$ and $\phi_w = \nu\tilde{n}_w.\sigma_w$, where $\tilde{n}_w = \tilde{n} \cup \{w\}$, $\sigma_s = \sigma \cup \{x \mapsto s\}$, and $\sigma_w = \sigma \cup \{x \mapsto w\}$. Here, ϕ_s represents the attacker's knowledge using the right guess, while ϕ_w represents his knowledge when his guess is wrong.

Guess verifiers. We consider two ways in which an attacker can verify whether his guess w is correct. First, he can use his guess to construct a pair of terms (t, t') that are equal under $\approx_R$ if $w = s$, but different if $w \neq s$. This is equivalent to ϕ_w and ϕ_s not being statically equivalent, and is the usual definition of security against off-line guessing used in symbolic methods [16, 20, 27]. Second, he can use his guess to construct a term t whose corresponding bitstring satisfies some given property if $w = s$, and not necessarily otherwise.

Given a term t and $p \in \mathbb{N}^*$, we denote the *subterm of t at position p* by $t|_p$, where $t|_\epsilon = t$ and, for $t = f(t_1, \ldots, t_n)$, $t|_{i.p} = t_i|_p$ for $i \in \{1, \ldots, n\}$, where $i.p$ denotes the sequence of integers obtained by prepending i to the sequence p. The set $eqv(\phi, t)$ of *equational verifiers* of a term t (under ϕ) is the set of pairs (t, t') such that $t, t' \in T_{\phi_s}$, $t\sigma_s \approx_R t'\sigma_s$, $t\sigma_w \not\approx_R t'\sigma_w$, and there is no $p \in \mathbb{N}^* \setminus \{\epsilon\}$ such that these conditions hold for the pair $(t|_p, t'|_p)$. These are the pairs of recipes that an attacker may use to validate his guess using the first strategy.

To model the second attacking strategy, we will consider a set $\mathcal{TT}$ of *test types* that model the attacker's ability to test whether a bitstring is in a given set. The set $tv(\phi, t)$ of *type verifiers* of t (under ϕ) is the set of pairs (t, TT) such that $t \in T_{\phi_s}$, $T \in \mathcal{TT}$, $P_{\mu_{ROM}}[\widehat{t\sigma_s} \in [\![TT]\!]] = 1$, $P_{\mu_{ROM}}[\widehat{t\sigma_w} \in [\![TT]\!]] \neq 1$, and there are no $p \in \mathbb{N}^* \setminus \{\epsilon\}$, $TT' \in \mathcal{TT}$ such that these conditions hold for $(t|_p, TT')$. Note that to model a realistic attacker one must choose test types such that $[\![T]\!]$ is efficiently decidable for all $T \in \mathcal{TT}$, .

Example 8. We will consider the test types odd, with $[\![\mathsf{odd}]\!]$ corresponding to the set of 1024-bit bitstrings that represent an odd number, so that $|[\![\mathsf{odd}]\!]| = 2^{1023}$, and nspf, with $[\![\mathsf{nspf}]\!]$ corresponding to the set of 1024-bit bitstrings representing numbers with no prime factors smaller than 10^6. We have $|[\![\mathsf{nspf}]\!]| \approx \prod_{p \in P_{10^6}} (p-1)/p \approx 1/24$, where P_i represents the set of prime factors smaller than i. These test types are used to model off-line guessing attacks in Section 4.2.

Our requirements on the sub-positions of verifiers prevent us from having infinite sets of spurious verifiers. For instance, let $\mathsf{h}^0(t) = t$ and $\mathsf{h}^{n+1}(t) = \mathsf{h}(\mathsf{h}^n(t))$ for each $n \in \mathbb{N}$, and let (t, t') be an equational verifier. Without this requirement, all pairs $(\mathsf{h}^i(t), \mathsf{h}^i(t'))$ for $i \in \mathbb{N}$ would be verifiers as well. However, if an attacker tests the pair $(t\sigma_w, t'\sigma_w)$, he cannot obtain more information by testing the pairs $(h^i(t)\sigma_w, h^i(t')\sigma_w)$, for $i > 0$.

In [25] we describe an algorithm for computing equational and type verifiers for any signature Σ and any convergent rewriting system R.

4.2 Off-line guessing examples on EKE

The EKE (Encrypted Key Exchange) protocol, proposed in [22], is designed to allow two parties to exchange authenticated information using a weak symmetric key. The authors show that naive versions of the protocol, while possibly symbolically secure, are nevertheless subject to off-line guessing attacks when implemented using RSA public keys. These examples illustrate that such attacks can result from implementation details that, while often trivial, are outside the scope of traditional symbolic methods.

We now show how our methods can be used to model and estimate the success probability of two such off-line guessing attacks. In both cases it is sufficient to consider the first step of the protocol. Probability calculations in this section rely on the setup specification of our running example and are performed automatically by our prototype implementation in less than one second.

Example 9. In the first step of this version of the protocol, an agent A samples a bitstring from $[\![\mathsf{random}]\!]$ represented by a term $r \in \Sigma_0$ such that $type(r) = \mathsf{random}$, and uses it to compute an RSA public key $\langle \mathsf{mod}(r), \mathsf{expn}(r) \rangle$. Then, A (symmetrically) encrypts this public key with a password s shared between A and the intended recipient B. To keep our analysis simple, we assume that the participants encrypt the modulus and the exponent separately and send them over the network as a pair of encryptions (instead of the encryption of the pair). Thus, this first message is represented by the term $\langle \{\!|\mathsf{mod}(r)|\!\}_s, \{\!|\mathsf{expn}(r)|\!\}_s \rangle$. See [22] for a full description of the protocol and its variants.

After observing this message in the network, the attacker's knowledge is given by $\phi = \nu\tilde{n}.\sigma$, where $\sigma = \{x_1 \mapsto \langle \{\!|\mathsf{mod}(r)|\!\}_s, \{\!|\mathsf{expn}(r)|\!\}_s \rangle\}$ and $\tilde{n} = \{r\}$. The relevant frames for the analysis of off-line guessing are $\phi_s = \nu\tilde{n}_w.\sigma_s$ and $\phi_w = \nu\tilde{n}_w.\sigma_w$, where $\tilde{n}_w = \tilde{n} \cup \{w\}$, $\sigma_s = \sigma \cup \{x_2 \mapsto s\}$, and $\sigma_w = \sigma \cup \{x_2 \mapsto w\}$.

There are no equational verifiers: $eqv(\phi, s) = \emptyset$. However, while it may be infeasible to check whether the modulus is the product of two primes, an attacker can use his guess w to decrypt the pair sent by A and test whether the result is a 1024-bit modulus without small prime factors and an odd exponent e. Thus,

$$tv(\phi, s) = \left\{ (\{\!|\pi_1(x_1)|\!\}_{x_2}^{-1}, \mathsf{nspf}), (\{\!|\pi_2(x_1)|\!\}_{x_2}^{-1}, \mathsf{odd}) \right\}.$$

We have $\widehat{\pi_1(x_1)\sigma_w} \in \mathcal{B}^{1024}$. Thus, $\{\!|\pi_1(\widehat{x_1})|\!\}_{x_2}^{-1} \sigma_w$ is sampled from $\mathcal{B}^{(769,1024)}$, and the probability that $\{\!|\widehat{\pi_1(x_1)}|\!\}_{x_2}^{-1}$ has 1024 bits is $\left|\mathcal{B}^{1024}\right| / \left|\mathcal{B}^{(769,1024)}\right| = 2^{1024} / \left|\sum_{i=769}^{1024} 2^i\right| \approx 1/2$. The probability that a 1024-bits bitstring is in $[\![\mathsf{nspf}]\!]$ is approximately $1/24$, and the probability that $\{\!|\pi_2(x_1)|\!\}_{x_2}^{-1}$ is odd is $1/2$. Therefore, each wrong guess satisfies the two type verifiers with probability

$$P_\mu \left[\{\!|\pi_1(\widehat{x_1})|\!\}_{x_2}^{-1} \sigma_w \in [\![\mathsf{nspf}]\!], \{\!|\pi_2(\widehat{x_1})|\!\}_{x_2}^{-1} \sigma_w \in [\![\mathsf{odd}]\!] \right] \approx \frac{1}{2} \cdot \frac{1}{24} \cdot \frac{1}{2} = \frac{1}{96}.$$

Since there are $2^{24} - 1$ wrong guesses, we estimate the probability of success of this off-line guessing attack as described above to be $\frac{1}{1+(2^{24}-1)/96} \approx 2^{-17.5}$, corresponding to the probability of picking the right guess from those which satisfy the equational and type verifiers.

Example 10. Consider the same setup as in Example 9, except that only the exponent of the RSA public key is encrypted in the first message. The authors of EKE note that the protocol is still vulnerable to off-line guessing attacks: Since the exponent of an RSA key is always odd, one can decrypt each encryption of a public key with each guess. For the right guess, decrypting each encryption will

yield an odd exponent. The probability that a wrong guess achieves this decreases exponentially with the number of encryptions available to the attacker [22].

To formalize this in our setting, we let $\phi = \nu\tilde{n}.\sigma$ be the frame representing the attacker's knowledge, where $\sigma = \{x_i \mapsto \langle \mathsf{mod}(r_i), \{\!|\mathsf{expn}(r_i)|\!\}_s \rangle \mid i \in \{1,\ldots,n\}\}$ and $\tilde{n} = \{r_1,\ldots,r_n,s\}$. The frames ϕ_s and ϕ_w used are as expected: $\phi_s = \nu\tilde{n}_w.\sigma_s$ and $\phi_w = \nu\tilde{n}_w.\sigma_w$, where $\tilde{n}_w = \tilde{n} \cup \{w\}$, $\sigma_s = \sigma \cup \{x_{n+1} \mapsto s\}$, and $\sigma_w = \sigma \cup \{x_{n+1} \mapsto w\}$. As before, there are no equational verifiers: $eqv(\phi, s) = \emptyset$. The set of type verifiers is given by $tv(\phi, s) = \left\{ (\{\!|\pi_2(x_i)|\!\}^{-1}_{x_{n+1}}, \mathsf{odd}) \mid i \in \{1,\ldots,n\} \right\}$. As in Example 9, we obtain $1/(1 + (2^{24}-1)/2^{n+1}) = 2^{n+1}/(2^{n+1} + 2^{24} - 1)$ as an estimate for the success probability of this off-line guessing attack.

We remark that when assessing the threat level of off-line guessing attacks one must consider not only the probability of success, but also the computational effort involved, i.e., the number of guesses that must be verified. In the attacks modeled by our method, this number is approximated by $G * p$, where G is the size of the space of guesses to be tried and p is the probability that a random guess satisfies all verifiers. This corresponds to the expected number of guesses that an attacker must try before finding one that satisfies all verifiers.

5 Conclusion

We presented a symbolic and automatable probabilistic framework for security protocol analysis. Our framework allows one to express properties of cryptographic primitives which are outside the scope of Dolev-Yao models, thereby modeling a stronger attacker. We illustrated its usefulness by modeling non-trivial properties of RSA and using them to analyze off-line guessing attacks on the EKE protocol which cannot be modeled by existing symbolic methods.

We have proposed a probability distribution based on interpreting functions as random oracles subject to satisfying the properties of cryptographic primitives described in our setup. This is a non-trivial generalization of the random oracle model. By using this probability distribution, we can (automatically) reason about an attack's success probability. In [28] we provide a prototype implementation of our methods, which computes probabilities in our formalization of a Dolev-Yao attacker using RSA asymmetric encryption and terminates in less than one second for all the examples presented in the paper.

More generally, our approach can be used to analyze a broad range of attacks and weaknesses of cryptographic primitives that could not previously be analyzed by symbolic models. These include some forms of cryptanalysis (such as differential cryptanalysis to AES, DES or hash functions, as in [29]) and side-channel attacks [24]. Short-string authentication, used in device pairing protocols, and distance-bounding protocols relying on rapid-bit exchange, are ill-suited for analysis with existing symbolic methods as their analysis is intrinsically probabilistic. However, they are amenable to analysis using our framework.

As future work, we plan to integrate this approach with a symbolic protocol model-checker capable of generating protocol execution traces and the probabilistic queries relevant for deciding whether a trace allows an attack. In the

case of off-line guessing, this amounts to computing the sets of equational and type verifiers, a task closely related to that of deciding static equivalence. Since our probabilistic analysis can be performed automatically (as illustrated by our prototype), this allows our analysis to be fully automated. We expect that such an approach will allow us to find numerous new protocol attacks relying on properties of the cryptographic primitives used.

Acknowledgments. The authors would like to thank Luís Alexandre Pereira, Pedro Adão, Benedikt Schmidt, Mohammad Torabi Dashti and Srdjan Marinovic for helpful discussions and feedback, as well as the anonymous reviewers for insightful comments and suggestions. Bruno Conchinha is a recipient of the Google Europe Fellowship in Computer Security, and this research is supported in part by this Google Fellowship. He also acknowledges the support of the FCT PhD grant SFRH/BD/44204/2008. Carlos Caleiro acknowledges the support of the project GeTFun EU FP7 PIRSES-GA-2012-318986, and the FEDER/FCT projects PEst-OE/EEI/LA0008/2013, ComFormCrypt PTDC/EIA-CCO/113033/2009 and UTAustin/MAT/0057/2008 AMDSC of IST.

References

1. V. Cortier, S. Delaune, and P. Lafourcade, "A survey of algebraic properties used in cryptographic protocols," *J. Comput. Secur.*, vol. 14, pp. 1–43, January 2006.
2. B. Blanchet, "An efficient cryptographic protocol verifier based on Prolog rules," in *Proc. of the 14th IEEE workshop on Computer Security Foundations*, CSFW '01, (Washington, DC, USA), pp. 82–96, IEEE Computer Society, 2001.
3. B. Schmidt, S. Meier, C. J. F. Cremers, and D. A. Basin, "Automated analysis of Diffie-Hellman protocols and advanced security properties," in *CSF* (S. Chong, ed.), pp. 78–94, IEEE, 2012.
4. S. Goldwasser and S. Micali, "Probabilistic encryption," *J. Comput. Syst. Sci.*, vol. 28, no. 2, pp. 270–299, 1984.
5. M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *CRYPTO* (D. R. Stinson, ed.), vol. 773 of *Lecture Notes in Computer Science*, pp. 232–249, Springer, 1993.
6. B. Blanchet, "Security protocol verification: Symbolic and computational models," in Degano and Guttman [30], pp. 3–29.
7. M. Abadi and P. Rogaway, "Reconciling two views of cryptography (the computational soundness of formal encryption)," *J. Cryptology*, vol. 20, no. 3, p. 395, 2007.
8. M. Backes, A. Malik, and D. Unruh, "Computational soundness without protocol restrictions," in *ACM Conference on Computer and Communications Security* (T. Yu, G. Danezis, and V. D. Gligor, eds.), pp. 699–711, ACM, 2012.
9. M. Backes, B. Pfitzmann, and M. Waidner, "A composable cryptographic library with nested operations," in *ACM Conference on Computer and Communications Security* (S. Jajodia, V. Atluri, and T. Jaeger, eds.), pp. 220–230, ACM, 2003.
10. H. Comon-Lundh and V. Cortier, "Computational soundness of observational equivalence," in *ACM Conference on Computer and Communications Security* (P. Ning, P. F. Syverson, and S. Jha, eds.), pp. 109–118, ACM, 2008.

11. B. Blanchet, "A computationally sound mechanized prover for security protocols," *IEEE Trans. Dependable Sec. Comput.*, vol. 5, no. 4, pp. 193–207, 2008.
12. G. Barthe, B. Grégoire, and S. Z. Béguelin, "Formal certification of code-based cryptographic proofs," in *POPL* (Z. Shao and B. C. Pierce, eds.), pp. 90–101, ACM, 2009.
13. G. Barthe, J. M. Crespo, B. Grégoire, C. Kunz, and S. Zanella Béguelin, "Computer-aided cryptographic proofs," in *3rd International Conference on Interactive Theorem Proving, ITP 2012*, pp. 12–27, Springer, 2012.
14. G. Bana and H. Comon-Lundh, "Towards unconditional soundness: Computationally complete symbolic attacker," in Degano and Guttman [30], pp. 189–208.
15. H. Comon-Lundh, V. Cortier, and G. Scerri, "Tractable inference systems: An extension with a deducibility predicate," in *CADE* (M. P. Bonacina, ed.), vol. 7898 of *Lecture Notes in Computer Science*, pp. 91–108, Springer, 2013.
16. M. Baudet, "Deciding security of protocols against off-line guessing attacks," in *Proceedings of the 12th ACM conference on Computer and communications security*, CCS '05, (New York, NY, USA), pp. 16–25, ACM, 2005.
17. R. Corin, J. Doumen, and S. Etalle, "Analysing password protocol security against off-line dictionary attacks," *Electron. Notes Theor. Comput. Sci.*, vol. 121, pp. 47–63, February 2005.
18. S. Halevi and H. Krawczyk, "Public-key cryptography and password protocols," *ACM Trans. Inf. Syst. Secur.*, vol. 2, no. 3, pp. 230–268, 1999.
19. M. Abadi and B. Warinschi, "Password-based encryption analyzed," in *ICALP*, vol. 3580 of *Lecture Notes in Computer Science*, pp. 664–676, Springer, 2005.
20. M. Abadi, M. Baudet, and B. Warinschi, "Guessing attacks and the computational soundness of static equivalence," *Journal of Computer Security*, pp. 909–968, December 2010.
21. "Sectools.org: Top 125 network security tools," Jan. 2013.
22. S. M. Bellovin and M. Merritt, "Encrypted Key Exchange: Password-based protocols secure against dictionary attacks," in *IEEE Symposium on Research in Security and Privacy*, pp. 72–84, 1992.
23. J. Munilla and A. Peinado, "Off-line password-guessing attack to Peyravian-Jeffries's remote user authentication protocol," *Computer Communications*, vol. 30, no. 1, pp. 52–54, 2006.
24. B. Köpf and D. A. Basin, "An information-theoretic model for adaptive side-channel attacks," in *CCS 2007*, pp. 286–296, ACM, 2007.
25. "`http://www.infsec.ethz.ch/people/brunoco/infsec/people/brunoco/esorics13_tech.pdf`," 2013.
26. M. Abadi and V. Cortier, "Deciding knowledge in security protocols under equational theories," *Theor. Comput. Sci.*, vol. 367, pp. 2–32, November 2006.
27. M. Abadi and C. Fournet, "Mobile values, new names, and secure communication," POPL '01, (New York, NY, USA), pp. 104–115, ACM, 2001.
28. "`http://www.infsec.ethz.ch/people/brunoco/prob_rsa.tar.gz`," 2013.
29. B. Montalto and C. Caleiro, "Modeling and reasoning about an attacker with cryptanalytical capabilities," *ENTCS*, vol. 253, no. 3, pp. 143–165, 2009.
30. P. Degano and J. D. Guttman, eds., *POST 2012*, vol. 7215 of *Lecture Notes in Computer Science*, Springer, 2012.