Pseudorandom Permutations and Equivalence of Formal Expressions (Abstract)

Peeter Laud*
Tartu University and Cybernetica AS
peeter@cyber.ee

1 Introduction and Problem Statement

Two different views of cryptography have developed over the years. In one of them — the formal view — the cryptographic operations are seen as functions over a space of formal expressions (modeled as elements of some term algebra), in the other one — the computational view — these operations are modeled as functions over bit-strings (which are the model for messages). Both views also use their own means to define the security of these operations — they are either determined by the structure of messages (for example, it is defined that an attacker seeing $\{E\}_K$ — "encryption of message E under key K" and K can also obtain E) or by the capabilities of algorithms running in probabilistic polynomial time (PPT). The computational view is arguably more realistic of the two, while the formal view allows simpler reasoning.

These two views have developed quite independently. Only in the last few years the question of their relationship has begun to be addressed. In [1], Abadi and Rogaway have shown that for a certain security definition of computational encryption — it must hide the identity of both plaintexts and keys — these two views are essentially the same for the passive attacker. They define an *interpretation* of formal expressions as bit-strings (it makes use of the computational encryption), and an equivalence relation over formal expressions (stated in terms of the formal model). They then show, that if two formal expressions are equivalent, then no PPT algorithm can tell their interpretations apart.

Encryption primitives hiding the identities of plaintexts and keys are not the most simple ones. They are constructed using simpler primitives which itself satisfy weaker security requirements. These primitives are *block ciphers* (for example, DES, Idea, Blowfish or AES) and they are required to be *pseudorandom permutations* (PRP). A block cipher is a triple of algorithms $(\mathcal{G}, \mathcal{E}, \mathcal{D})$, where $\mathcal{G}$ is the *key generation*, $\mathcal{E}$ is the *encryption*, and $\mathcal{D}$ is the *decryption* algorithm. They all take as an argument the *security parameter* $n \in \mathbb{N}$. This parameter determines the length of keys and plain- and ciphertexts, we let them both be equal to $\ell(n)$, where ℓ is a polynomial. Additionally, both $\mathcal{E}$ and $\mathcal{D}$ take a key — a bit-string of length $\ell(n)$ as an argument. The last argument of $\mathcal{E}$ is a plain-

*Supported by Estonian Science Foundation grant #5279

text and the last argument of $\mathcal{D}$ is a ciphertext, both of them are bit-strings of length $\ell(n)$. The algorithm $\mathcal{G}$ is probabilistic — on invocation it returns a new key. The algorithms $\mathcal{E}$ and $\mathcal{D}$ are deterministic. A block cipher is a PRP, if no PPT algorithm has a significant advantage (the advantage must be at least the reciprocal of some polynomial over n) over simple guessing in telling a randomly chosen encryption function $\mathcal{E}(n, k, \cdot)$ (where k is distributed according to $\mathcal{G}(n)$) apart from a uniformly randomly chosen permutation of the set of bit-strings of length $\ell(n)$. We see that a PRP does not hide the identities of plaintexts and keys (for hiding the identity of plaintexts, the encryption must be probabilistic).

As the pseudorandom permutations are truly primitive operations, one may ask, how should the equivalence of formal expressions look like, if the computational encryption, used by the interpretation of expressions, is just a PRP. This is the topic of our contribution.

2 The Analysis

First we have to specify, what are the elements of our set of formal expressions **Exp**. First, they contain all the elements K of the countable set $\mathbf{K}$ of *formal keys.* Second, if $E \in \mathbf{Exp}$ and $K \in \mathbf{K}$, then also $\{E\}_K \in \mathbf{Exp}$, we have already described its meaning. Third, we assume that there is a finite set **Op** of *formal operations* o, each one with a fixed arity $\mathrm{ar}(o)$. If $E_1, \ldots, E_{\mathrm{ar}(o)} \in \mathbf{Exp}$, then also $o(E_1, \ldots, E_{\mathrm{ar}(o)}) \in \mathbf{Exp}$. One operation that we need later is the pairing $(\cdot, \cdot)$. The operations in **Op** may denote usual arithmetic operations like addition or multiplication. Last, we also let *formal inputs* $i \in \mathbf{Inp}$, where **Inp** is a countable set, to be elements of **Exp**. Formal inputs are not the same as nullary operations, they are interpreted differently.

The computational interpretation $[\![E]\!]$ of a formal expression E is the following. Fix a PRP $(\mathcal{G}, \mathcal{E}, \mathcal{D})$. For a fixed $n \in \mathbb{N}$, the keys $K \in \mathbf{K}$ are interpreted by invoking $\mathcal{G}(n)$ (one invocation for one key). For each $o \in \mathbf{Op}$ we have fixed is *semantics* — a deterministic algorithm $[\![o]\!]$ that takes the security parameter n and $\mathrm{ar}(o)$ bit-strings and returns a bit-string. The expressions $o(E_1, \ldots, E_k)$ are interpreted by invoking that algorithm. We assume that the interpretation of the pairing is an injective function τ from pairs of bit-strings to bit-strings. The expressions $\{E\}_K$ are interpreted in the same way, but here the used algorithm is $\mathcal{E}$. Finally, the formal inputs $i \in \mathbf{Inp}$ are interpreted as bit-strings, and there is a fixed probability distribution over the mappings of elements of **Inp** to bit-strings.

The next step is defining a language of *claims* **CL**. A claim takes some formal expressions as its arguments and states something about their computational interpretations. This statement may be either true or false, depending on the semantics of operations and the distribution of inputs. For example, for each two formal expressions there is a claim $\mathsf{SD}(E_1, E_2)$ stating that the computational interpretations of these two expressions are indistinguishable (for a PPT algorithm). After defining the language **CL**, we give a number of implications of the form "if claims $C_1, \ldots, C_k$ hold then the claim C_{k+1} also holds". In general, the expressions in $C_1, \ldots, C_k$ are simpler than the expressions in C_{k+1}. Starting from the claims describing the distribution of inputs, we can derive with help of these implications statements about the formal expressions that we are interested in. This is our analysis. The claims $\mathsf{SD}(E_1, E_2)$, whose validity we can derive, constitute the equivalence relation we mentioned at the end of the previous section.

The claims in **CL** are the following:

- $\mathsf{R}(E)$, this claim is true iff the distribution $[\![E]\!]$ lays only negligible weight on each particular bit-string.
- $\mathsf{U}(E)$, this claim is true iff the distribution $[\![E]\!]$ is indistinguishable from the uniform distribution over bit-strings of length $|[\![E]\!]|$.
- $\not\equiv(E_1, E_2)$, this claim is true, iff the probability that the interpretations of E_1 and E_2 are equal, is negligible.
- $\mathsf{Ind}(\{E_1, \ldots, E_k\}, \{E'_1, \ldots, E'_l\})$, this claim is true iff the interpretation of $(E_1, \ldots, E_k)$ is *independent* of the interpretation of $(E'_1, \ldots, E'_l)$. More concretely, the distribution

  $$[\![((E_1, \ldots, E_k), (E'_1, \ldots, E'_l))]\!]$$

  must be indistinguishable from the distribution

  $$\tau([\![(E_1, \ldots, E_k)]\!], [\![(E'_1, \ldots, E'_l)]\!]) \ .$$

- $\mathsf{SD}(\{(E_1, E'_1), \ldots, (E_k, E'_k)\})$, this claim is true iff the distributions $[\![(E_1, \ldots, E_k)]\!]$ and $[\![(E'_1, \ldots, E'_k)]\!]$ are indistinguishable.
- $\mathsf{NE}(K, E, E')$, this is just a shorthand for "for all subexpressions $\{E''\}_K$ of E', the claim $\not\equiv(E, E'')$ holds".

Let $X, Y \subset \mathbf{Exp}$. Some more interesting implications are:

- If $E_1, \ldots, E_k \in X$, then $\mathsf{Ind}(X, Y) \Rightarrow \mathsf{Ind}(X \cup \{o(E_1, \ldots, E_k)\}, Y)$.
- $\mathsf{Ind}(X, Y) \wedge \bigwedge_{E' \in X \cup Y \cup \{E\}} \mathsf{NE}(K, E, E') \Rightarrow \mathsf{Ind}(X \cup \{\{E\}_K\}, Y)$.
- $\mathsf{NE}(K, E, E) \Rightarrow \mathsf{U}(\{E\}_K)$.
- $\mathsf{NE}(K, E, E') \Rightarrow \not\equiv(\{E\}_K, E')$.
- $\mathsf{R}(E) \wedge \mathsf{Ind}(\{E\}, \{E'\}) \Rightarrow \not\equiv(E, E')$.
- $\mathsf{Ind}(\{E_i\}_{i=1}^k, \{F_i\}_{i=1}^l) \wedge \mathsf{SD}(\{(E_i, E'_i)\}_{i=1}^k) \wedge \mathsf{SD}(\{(F_i, F'_i)\}_{i=1}^l) \wedge$ $\mathsf{Ind}(\{E'_i\}_{i=1}^k, \{F'_i\}_{i=1}^l) \Rightarrow \mathsf{SD}(\{(E_i, E'_i)\}_{i=1}^k \cup \{(F_i, F'_i)\}_{i=1}^l)$.

Our analysis is precise enough to automatically derive the security of some block-ciphers' modes of operation (see [3, Chap. 9]). We can derive that the pairs of plaintext and corresponding ciphertext are indistinguishable from pairs of a plaintext and a random bit-string.

Full analysis (together with proofs) is presented in [2, Chap. 6].

References

[1] Martín Abadi and Phillip Rogaway. Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption). In Jan van Leeuwen, Osamu Watanabe, Masami Hagiya, Peter D. Mosses, and Takayasu Ito, editors, *International Conference IFIP TCS 2000*, volume 1872 of *LNCS*, pages 3–22, Sendai, Japan, August 2000. Springer-Verlag.

[2] Peeter Laud. *Computationally Secure Information Flow.* PhD thesis, Universität des Saarlandes, 2002.

[3] Bruce Schneier. *Applied Cryptography; Protocols, Algorithms, and Source Code in C.* John Wiley & Sons, 1996.