SAND2008-3520C

On the Secure Obfuscation of Deterministic Finite Automata

W. Erik Anderson

Sandia National Laboratories* Albuquerque, NM 87185-0785
weander@sandia.gov

Abstract. In this paper, we show how to construct secure obfuscation for Deterministic Finite Automata, assuming non-uniformly strong one-way functions exist. We revisit the software protection approaches originally proposed by [5, 8, 10, 13] and revise them to the current obfuscation setting of Barak et al. [2]. Under this model, we introduce an efficient oracle that retains some "small" secret about the original program. Using this secret, we can construct an obfuscator and two-party protocol that securely obfuscates Deterministic Finite Automata against *malicious* adversaries. The security of this model retains the strong "virtual black box" property originally proposed in [2] while incorporating the stronger condition of dependent auxiliary inputs in [12].

Keywords: Obfuscation, deterministic finite automata, state machines, authenticated encryption, oracle machines, provable security, game-playing.

1 Introduction

Program obfuscation, if possible and practical, would have a considerable impact on the way we protect software systems today. It would be instrumental in protecting intellectual property, preventing software piracy, and managing use-control applications. The work of Barak et al. [2] initiated the first formal study of obfuscation. They define an obfuscator $\mathcal{O}$ to be an efficient, probabilistic compiler that takes a program P and transforms it into a functionally equivalent yet unintelligible program $\mathcal{O}(P)$. Unintelligible is defined in the strictest sense, to imply that the program $\mathcal{O}(P)$ behaves ideally like a "virtual black box". That is, whatever can be efficiently extracted from the obfuscated program can also be extracted efficiently when given only oracle access to P.

Unfortunately, in [2] it was proven that obfuscation in general is impossible. Namely, there exist a family of functions that are unobfuscatable under the "virtual black box" notion of security. This would seem to suggest that having physical access to the program is a much stronger capability than having only access to its input and output behavior. In addition to this main impossibility result, the authors also prove that if secure symmetric key encryption schemes exist, pseudorandom functions exist, or message authentication schemes exist, then so do unobfuscatable versions of each. The authors conclude that the "virtual black box" property is inherently flawed, and if we hope for any positive obfuscation results, then either this model needs to be abandoned or we need to accept that many programs are not obfuscatable [2].

Numerous other impossibility results have also shed light on the problem of obfuscation. For example, Goldwasser et al. [12] showed that many natural circuit classes are unobfuscatable when auxiliary inputs are added to the obfuscation model. Auxiliary inputs provide for a more robust model of obfuscation, since the adversary is assumed to have some a priori information about the

* Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

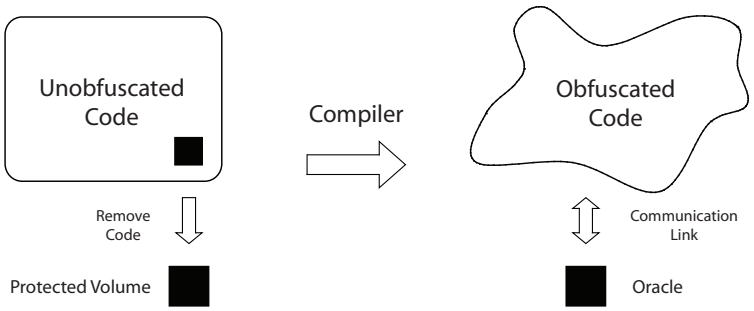

Fig. 1. Obfuscation with respect to oracle machines

underlying program. This additional layer of security is useful in practice, since it is likely that the obfuscated code will be utilized in a large system, and the system may reveal partial information about the functionality of the obfuscated code.

In spite of the numerous impossibility results, other works such as Lynn et al. [14] have examined alternative models of obfuscation in the hope of achieving meaningful possibility results. Under the random oracle model of obfuscation, they assume that both the obfuscator and obfuscated code have access to a public random oracle. Under this assumption, they are able to show that both point functions and complex access control schemes are obfuscatable. Similar results were obtained by [6, 7, 15] under a slightly weaker notion of "virtual black box" obfuscation (without random oracles). For example, Wee showed in [15] that point functions are (weakly) obfuscatable, provided that strong one-way permutations exist.

In this paper, we introduce a new model of obfuscation that has wide and meaningful possibility results beyond those described above. To demonstrate the utility of this model, we show that deterministic finite automata are securely obfuscatable, provided non-uniformly strong one-way functions exist. In fact with a simple modification these same results can be extended to the obfuscation of Turing machines and hence all programs. For a more detailed discussion on this extension along with composition results see the extended abstract [1]. We call this model of obfuscation *obfuscation w.r.t. oracle machines*.

Unlike the "virtual black box" model of obfuscation, where we assume an adversary has full access to the obfuscated code, we instead consider the case where a small portion of the computation remains hidden and is only accessible via a black box oracle. See Figure 1 for an illustration. A compiler in this case takes a program P and returns two outputs, the obfuscated code $\mathcal{O}(P)$ which is given to the adversary, and a small secret which is given to the oracle. An execution of the obfuscated code takes an input x and computes $\mathcal{O}(P)(x)$, via a two-party protocol. To avoid certain trivialities, we impose restrictions on the oracle's computational resources. Namely, we will consider only the case when the oracle's space resources are asymptotically smaller than the program itself. In practice, the oracle may be implemented as a computationally limited device, such as a smart card or crypto processor.

1.1 Notation

We will use the notation PPT to stand for probabilistic polynomial-time Turing machine. If A is a PPT, B an oracle, and x an input to A, then by $A^B(x)$ we mean the algorithm that runs on

input x using oracle access to B. We will often refer to A as a PPT oracle machine. When writing $x \Leftarrow A$ we mean the value x is returned by A. Additionally, $A(1^k)$ implies A is given the value k. In our algorithm descriptions, we make use of the statements **return** y and **Return** z. When using the syntax **return**, we imply that the value y is returned internally to the algorithm (such as the output of a function call), while by using **Return**, we imply that the value z is written to the output tape. As usual, we use the notation $\{0,1\}^k$ to denote the set of all k-bit binary strings, and by $x \xleftarrow{\$} \{0,1\}^k$ we mean x is uniformly chosen from $\{0,1\}^k$. We also use the conventional notation of $\|$ and $\oplus$ to denote the string operators *concatenate* and *exclusive or*. Unless explicitly stated otherwise, we will assume all references to log are based 2. A function $\mu : \mathbb{N} \to \mathbb{R}^+$ is said to be *negligible*, if for any positive polynomial p there exists an integer N such that for any $k > N$, $\mu(k) < 1/p(k)$. We will sometimes use the notation $neg(\cdot)$ to denote an arbitrary negligible function. A polynomial-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ is called non-uniformly strong one-way if for every non-uniform PPT A there is a negligible function $neg(\cdot)$ such that for sufficiently large k, $\Pr_{x \xleftarrow{\$} \{0,1\}^k}[\, f^{-1}(f(x)) \ni y \leftarrow A(f(x), 1^k)\,] \leq neg(k)$.

2 Obfuscation with respect to Oracle Machines

In this section we introduce the framework for *obfuscating w.r.t. oracle machines*. Under this framework we model obfuscation as a two-party protocol, where one party represents the obfuscated code and the other an oracle containing some "small" secret. The communication between the two parties is characterized using *interactive Turing machines* introduced by [11]. Under this model, we assume the adversary has complete control over both the obfuscated code and message scheduling. We further assume the adversary is *malicious*, and may deviate from the protocol in any way. This allows the adversary to adaptively query the oracle with messages of its own choice.

Oracle Model. The obfuscation oracle $\mathcal{R}$ is modeled as an interactive Turing machine with one additional read-and-write tape called *internal_state*. The tape *internal_state* has a unique feature called *persistence* that distinguishes it from the other tapes in the oracle. We say a tape is *persistent* if the tape's contents are preserved between each successive execution of $\mathcal{R}$. The other internal working tapes do not share this property and are assumed to be blank after each execution. Given a particular *input* and *internal_state*, the oracle $\mathcal{R}$ computes an *output* (which may be $\perp$) on its outgoing communication tape along with a new internal state, *internal_state*$'$.

$$(output, internal\_state') \leftarrow \mathcal{R}(input, internal\_state)$$

We will assume that the oracle's input and persistent tape are polynomial bounded by some polynomial $s(k)$, for each $k \in \mathbb{N}$. In this framework, we will consider only the non-trivial case when $s(k) = o(f(k))$,[1] where the device's resources are asymptotically smaller than the program itself. In the special case when $s(k) = O(k)$, we will say that the oracle maintains a "small" internal state.

Definition 1. (Obfuscation w.r.t. Oracle Machines) *A probabilistic polynomial time algorithm $\mathcal{O}$ and oracle $\mathcal{R}$ are said to comprise an obfuscator of the family $\mathcal{F} = \{\mathcal{F}_k\}_{k \in \mathbb{N}}$ w.r.t. polynomial-time bounded oracle machines, if the following three conditions hold:*

[1] The size of each $M \in \mathcal{F}_k$ is polynomial bounded by $f(k)$.

- *(Approximate Functionality) There exists a negligible function μ such that for all k and $M \in \mathcal{F}_k$, $\mathcal{O}^{\mathcal{R}}(M, 1^k)$ describes an ITM that computes the same function as M with probability at least $1 - \mu(k)$.*
- *(Polynomial Slowdown) The description length and running time of $\mathcal{O}^{\mathcal{R}}(M, 1^k)$ is at most polynomial larger than that of M. That is, there exists a polynomial p such that for all k and $M \in \mathcal{F}_k$, $|\mathcal{O}(M, 1^k)| \leq p(k)$ and if M takes t time steps on an input x, then $\mathcal{O}^{\mathcal{R}}(M, 1^k)$ takes at most $p(k + t)$ time steps on x.*
- *(Virtual Black Box) For every PPT A, there is a PPT simulator S and a negligible function ν such that, for all k and $M \in \mathcal{F}_k$, and every polynomial q with bounded auxiliary input z of size $q(k)$, we have*

$$\left|\Pr[A^{\mathcal{R}}(\mathcal{O}^{\mathcal{R}}(M, 1^k), 1^k, z) = 1] - \Pr[S^M(1^{|M|}, 1^k, z) = 1]\right| \leq \nu(k).$$

2.1 Non-Resettable Deterministic Finite Automata

We define a *Deterministic Finite Automaton* (DFA) as a machine $\Psi = (Q, \Sigma, \delta, s_0, G)$ with a finite set of states Q, finite alphabet Σ, transition function δ, initial state $s_0 \in Q$, and accepting states G. The structure of the DFA is determined by its transition function δ, which maps each state and a given input symbol to a new state. The output function (which imitates black box behavior) of the DFA Ψ is defined as

$$\Psi(s, \alpha) := \begin{cases} 1 & \text{if } \delta(s, \alpha) \in G \\ 0 & \text{if } \delta(s, \alpha) \notin G \end{cases}$$

where the "user"-selectable input is α, and s is the current "internal" state. We note that the user does not have control of the state input. Rather Ψ must internally maintain the state over each execution. We will often write just $\Psi(\alpha)$.

When modeling DFAs, it is often convenient (unless stated otherwise) to assign a *reset* capability, which allows the DFA to transition back to its initial state. In practice, having a reset capability is not always a desired characteristic, especially when developing software use-control applications, such as subscription policies and digital rights management. To differentiate between DFAs that have a reset capability and those that don't, we define a *non-resettable* DFA to be a deterministic finite automaton that is not resettable. We note that we can always build in resettability if we add an additional reset symbol to every state.

3 DFA Obfuscation

Following the framework described in Section 2, we show how to construct a DFA obfuscator that is secure with respect to dependent auxiliary inputs. Our goal is to develop a compact, yet very efficient DFA obfuscator that is not only of theoretical interest, but useful in practice as well. To obtain our results, we use a simple authenticated encryption scheme to hide the structure of the DFA and authenticate the execution of the protocol.

Representation. We model each DFA Ψ as a polynomial-time Turing machine M_Ψ with an additional *persistent* read-and-write tape, called *internal_state*. The *internal_state* maintains a record of the values needed to compute the DFA. Each M_Ψ is represented by a table where, $\forall\ \alpha \in \Sigma$, $\forall\ s \in Q$, there is a table entry containing α, s, $\delta(s, \alpha)$, and *acpt* (which equals 1 iff $\delta(s, \alpha) \in G$). Without

any loss of functionality, we compress the table by employing an injective map that encodes each $\alpha \in \Sigma$ to a string in $\{0,1\}^{\lceil \log |\Sigma| \rceil}$. Using the table described, we can create a program M_Ψ that simulates Ψ's output behavior. The program consists of the DFA table, high-level code, and two persistent variables *current_state* and *current_acpt*. The high-level code describes the programming language used, table lookup algorithm, alphabet Σ, and function calls that manipulate the persistent variables. The program M_Ψ works as follows: On user input α, the table lookup algorithm searches each table entry for the pair α, *current_state*. If a match is found, the *acpt* bit is updated and $\delta(\textit{current\_state}, \alpha)$ is recorded temporarily. The program continues to search the rest of the table for a match. At the end of the table search, the user is given the recorded *acpt* bit, and the variable $\textit{current\_state} \leftarrow \delta(\textit{current\_state}, \alpha)$ is updated. After the *acpt* bit has been returned, the DFA is ready to accept its next input.

Following this description, our next goal is to define an encoding scheme of M_Ψ. Our choice of encoding is important for several reasons. First, it allows us to calculate the size of $|M_\Psi|$, which is needed for evaluating the polynomial slowdown property. And second, depending on our choice of encoding, the size of $|M_\Psi|$ may drastically affect the simulator's ability to simulate the obfuscated code. We formalize our encoding scheme as follows.

Encoding. We begin our encoding by splitting up the description of M_Ψ into its individual components: high-level code and DFA table (which is further broken down by individual table entries). We create a parsing scheme that takes the bit description of each component and adds a trailing bit of a 1 or 0 to the end of each individual bit. The trailing bit allows the parser to recognize the end of a component's description. For example if the high-level code has a bit description $h_0 \dots h_m$ then its new bit description is $h_0 0 h_1 0 \dots h_m 1$. Adopting this encoding scheme, we can find a $t \geq 0$ such that the size of each table entry satisfies $2^t \leq |\textit{table entry}| < 2^{t+1}$. Given t, we pad each table entry with the string $00 \dots 01$ (which is a multiple of two in length) until its length is exactly 2^{t+1}. If the number of tables entries is even, we pad the last table entry with an additional 2^{t+1} bits of the form $00 \dots 01$ and add a single 1 bit value on the end. If, on the other hand, the number of table entries is already odd, then we do nothing. For convenience we denote the number of edges in Ψ as $|E(\Psi)|$. By prefixing the parser to the encoded M_Ψ, it follows that $|M_\Psi| = |\text{Parser}| + |\text{High-level code}| + |\text{Table}|$, where $|\text{Table}| = 2^{t+1}|E(\Psi)|$ if the number of table entries is odd and $2^{t+1}(|E(\Psi)| + 1) + 1$ if the number of table entries is even.

Since both the size of the parser and the high-level code are public, it follows that knowing the size of $|M_\Psi|$ implies that one also knows the size of |Table|. But one can efficiently extract the number of edges $|E(\Psi)|$ based on our encoding above. We use this deduction later in the proof of Proposition 1 to swap the simulator's input $1^{|M_\Psi|}$ with $1^{|E(\Psi)|}$.

Based on the encoding scheme above, we define the family $\mathcal{F}_{\text{DFA}} := \{\mathcal{F}_k\}_{k \in \mathbb{N}}$ to be the set of all polynomial bounded M_Ψ satisfying

$$\mathcal{F}_k := \{M_\Psi \mid |M_\Psi| \leq f(k) \text{ and } 2\log|\textit{States}(\Psi)| + \log|\Sigma| + 1 < k\}$$ [2]

[2] The condition $2\log|\textit{States}(\Psi)| + \log|\Sigma| + 1 < k$ may be removed by modifying the encryption scheme in Figure 3 to have more than one call to F_K per table entry. This is a relatively easy fix since we need at most $m = \lceil (2t\log(ck) + 1)/k \rceil$ constant calls to F_K given $|M_\Psi| \leq f(k) \leq ck^t$ some fixed $c, t > 0$. This condition was added to simplify the obfuscation algorithm.

Setup(M_Ψ, k)**:**

INPUT: $M_\Psi, 1^k$

KEY GENERATION:
$K \leftarrow \mathcal{K}(k)$

GENERATE STATE TABLE:
STATETABLE(Ψ) :
$s \leftarrow 0$
$|m|^* \leftarrow \lceil \log_2 |\Sigma| \rceil + 2\lceil \log_2 |Q| \rceil + 1$
for $state \leftarrow 0$ **to** $|Q| - 1$ **do**
  for $symbol \leftarrow 0$ **to** $|\Sigma| - 1$ **do**
    $s_\alpha \leftarrow \alpha_{symbol}$
    $s_{state} \leftarrow state$
    $s_{\delta(state,\alpha)} \leftarrow \delta(state, \alpha_{symbol})$
    $s_{acpt} \leftarrow 1$ iff $s_{\delta(state,\alpha)} \in G$, 0 else
    $\mathbb{T}^*_{state}[s] \leftarrow$
      $s_\alpha \| s_{state} \| s_{\delta(state,\alpha)} \| s_{acpt} \| 0^{k-|m|^*}$
    $s \leftarrow s + 1$
$|Table|^* \leftarrow s$
return $(|m|^*, |Table|^*, \mathbb{T}^*_{state})$

ENCRYPT STATE TABLE ENTRIES:
$\mathcal{E}_{F_K}(\mathbb{T}^*_{state})$:
$X_1 \leftarrow 1^k$
$Auth \leftarrow F_K(X_1)$
for $s \leftarrow 0$ **to** $|Table|^* - 1$ **do**
  $X_0 \leftarrow s \| 0$
  $Y \leftarrow F_K(X_0)$
  $\mathbb{T}^*_C[s] \leftarrow Y \oplus \mathbb{T}_{state}[s]$
  $X_1 \leftarrow Auth \oplus \mathbb{T}^*_C[s]$
  $Auth \leftarrow F_K(X_1)$
$Auth^* \leftarrow Auth$
return $\mathbb{T}^*_C \| Auth^*$

Return $(K, |m|^*, |Table|^*, \mathbb{T}^*_C, Auth^*)$

Algorithm $\mathcal{O}(|Table|^*, \mathbb{T}^*_C, Auth^*)$:

INPUT: $|Table|^*, \mathbb{T}^*_C, Auth^*$

INITIALIZATION:
$|Table| \leftarrow |Table|^*$
$\mathbb{T}_C \leftarrow \mathbb{T}^*_C$
$Auth \leftarrow Auth^*$
$State \leftarrow$ **Transition_Query**

STATE TRANSITIONS:
Case$(State)$

Transition_Query:
$\alpha \leftarrow scan\_input$
Query oracle $\mathcal{R}$ with α
$State \leftarrow$ **State_Update**

State_Update:
for $s \leftarrow 0$ **to** $|Table| - 1$ **do**
  if $s \neq |Table| - 1$ **then**
    Query oracle $\mathcal{R}$ with $\mathbb{T}_C[s]$
  if $s = |Table| - 1$ **then**
    Query oracle $\mathcal{R}$ with $\mathbb{T}_C[s] \| Auth$

if $acpt \Leftarrow \mathcal{R}$
  Return $acpt$
  $State \leftarrow$ **Transition_Query**
if $auth\_fail \Leftarrow \mathcal{R}$
  $State \leftarrow$ **Transition_Query**

Fig. 2. Algorithm **Setup** and $\mathcal{O}$.

for some fixed polynomial $f(k)$. The parameter k is called the *security parameter*.

Obfuscation. To simplify our description of the DFA obfuscator, we split the obfuscation into three separate algorithms, Setup, $\mathcal{O}$, and $\mathcal{R}$. The Setup algorithm, shown in Figure 2, takes a DFA encoding M_Ψ and generates inputs for both the obfuscated code and oracle. Without loss of generality, we view our encoding of M_Ψ to be the DFA state transition table of Ψ. The parsing operation and high-level code was left out for simplicity.

The obfuscated code $\mathcal{O}$, also shown in Figure 2, can be described as a protocol template. The template takes as input the encrypted table $\mathbb{T}_C$, authentication tag $Auth$, and table size $|Table|$ returned by the Setup algorithm. During the **Transition_Query** phase the obfuscated code scans in the user's input α, queries the oracle $\mathcal{R}$, and enters a new phase called **State_Update**.

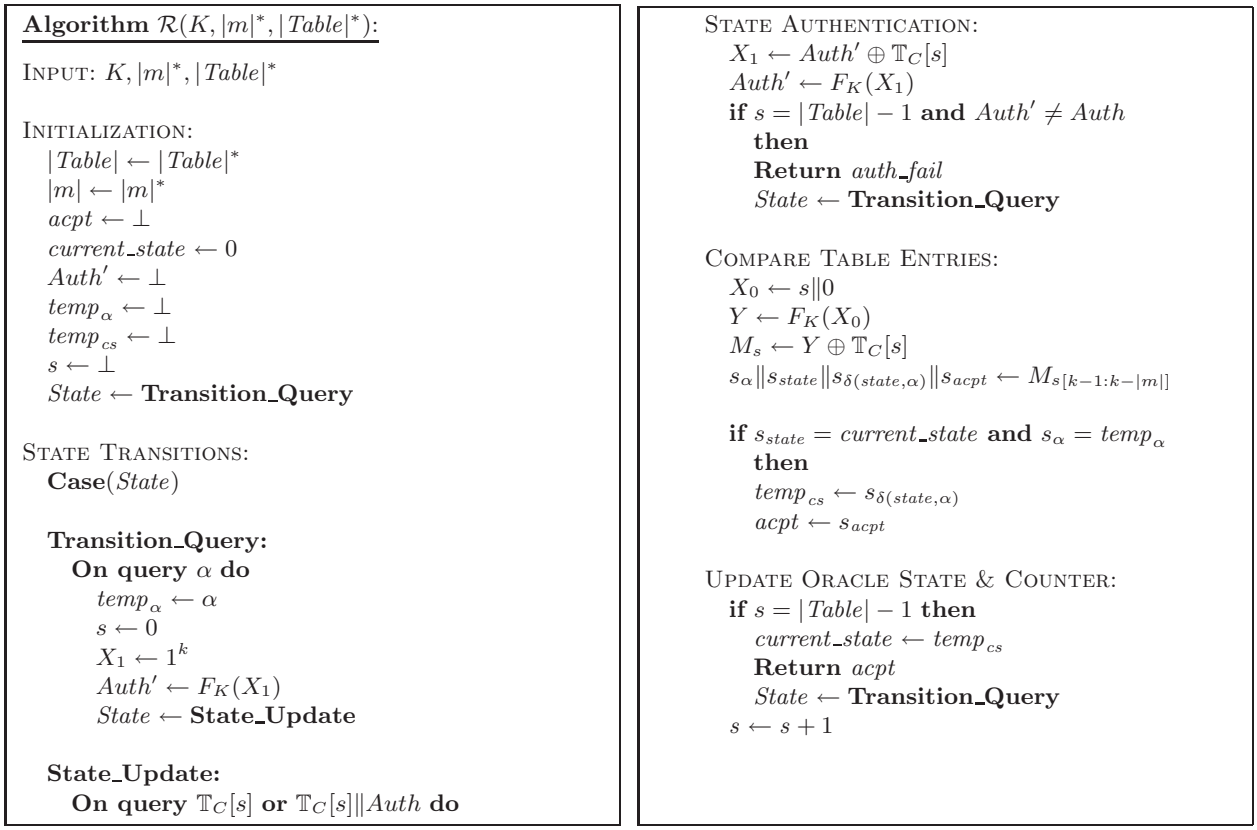

Fig. 3. Oracle $\mathcal{R}$.

During **State_Update**, the obfuscated code submits the table $\mathbb{T}_C$ along with the authentication tag $Auth$. The oracle processes $\mathbb{T}_C$ one table entry at a time and verifies the table's integrity. If the authentication passes, the oracle returns an accept value corresponding to whether the new state is an accept state.

The oracle $\mathcal{R}$, shown in Figure 3, describes the oracle's behavior. Just like the obfuscated code $\mathcal{O}$, the oracle is nothing more than a protocol with a symmetric key and a few additional variables. Other than the padding length $|m|$, table size $|Table|$, and $current\_state$, the oracle maintains no information about the DFA.

Proposition 1 *If non-uniformly strong one-way functions exist, then non-resettable DFAs are obfuscatable with respect to oracle machines.*

Proof: Let $f(k)$ be some positive polynomial and consider the family $\mathcal{F}_{\text{DFA}}$ defined over $f(k)$. We will assume without loss of generality for the remainder of the proof that k is sufficiently large so that the inequality $2\log|States(\Psi)|+\log|\Sigma|+1 < k$ is satisfied for every $M_\Psi \in \mathcal{F}_k$. This assumption follows from the fact that every $M_\Psi \in \mathcal{F}_k$ is polynomial bounded, and therefore there exists a fixed $t > 0$ with $|M_\Psi| \leq k^t$ for every k sufficiently large. Thus $|States(\Psi)||\Sigma| < |M_\Psi| \leq k^t$ implies $\log|States(\Psi)| + \log|\Sigma| < t\log k$ whence $2\log|States(\Psi)| + \log|\Sigma| + 1 \leq 2t\log k + 1 < k$ for k sufficiently large. This last restriction was added to guarantee that the size of each table entry is no larger than the size of the pseudorandom function's output length.

To prove that the obfuscator in Figure 2 and 3 obfuscates non-resettable DFAs, we need to show that the aforementioned three conditions hold: *Approximate Functionality*, *Polynomial Slowdown*, and *Virtual Black Box*. The *Approximate Functionality* and *Polynomial Slowdown* conditions are fairly straightforward and are left out in this proof. For details see the extended abstract.

Virtual Black Box: To simplify the notation in the proof we omit the input 1^k. We also replace the simulator input $1^{|M_\Psi|}$ with $1^{|E(\Psi)|}$ which can be extracted (based on our encoding of M_Ψ). This reduces the virtual black box inequality to Equation (1).

We begin our analysis by breaking up Equation (1) into four separate equations, each equation representing the indistinguishability of obfuscating with different oracles. Other than the first oracle $\mathcal{R}_{F_K}$ we do not place any computational assumptions on the others. This allows them to maintain a much larger internal state.

$$\left|\Pr[A^{\mathcal{R}_{F_K}}(\mathcal{O}^{\mathcal{R}_{F_K}}(\Psi), z) = 1] - \Pr[S^{\Psi}(1^{|E(\Psi)|}, z) = 1]\right| \tag{1}$$

$$\leq \left|\Pr[A^{\mathcal{R}_{F_K}}(\mathcal{O}^{\mathcal{R}_{F_K}}(\Psi), z) = 1] - \Pr[A^{\mathcal{R}_{\mathrm{Fun}}}(\mathcal{O}^{\mathcal{R}_{\mathrm{Fun}}}(\Psi), z) = 1]\right| \tag{2}$$

$$+ \left|\Pr[A^{\mathcal{R}_{\mathrm{Fun}}}(\mathcal{O}^{\mathcal{R}_{\mathrm{Fun}}}(\Psi), z) = 1] - \Pr[A^{\mathcal{R}^*_{\mathrm{Fun}}}(\mathcal{O}^{\mathcal{R}^*_{\mathrm{Fun}}}(\Psi), z) = 1]\right| \tag{3}$$

$$+ \left|\Pr[A^{\mathcal{R}^*_{\mathrm{Fun}}}(\mathcal{O}^{\mathcal{R}^*_{\mathrm{Fun}}}(\Psi), z) = 1] - \Pr[A^{\mathcal{R}^*_{\mathrm{Rand}}}(\mathcal{O}^{\mathcal{R}^*_{\mathrm{Rand}}}(\Psi), z) = 1]\right| \tag{4}$$

$$+ \left|\Pr[A^{\mathcal{R}^*_{\mathrm{Rand}}}(\mathcal{O}^{\mathcal{R}^*_{\mathrm{Rand}}}(\Psi), z) = 1] - \Pr[S^{\Psi}(1^{|E(\Psi)|}, z) = 1]\right|. \tag{5}$$

In Equation (2) we introduce the oracle $\mathcal{R}_{\mathrm{Fun}}$ in order to measure the pseudorandomness of $\mathcal{R}_{F_K}$. Both $\mathcal{R}_{\mathrm{Fun}}$ and $\mathcal{R}_{F_K}$ have the same description, except every call to F_K in $\mathcal{R}_{F_K}$ is replaced with a similar call to a random function (independent of z) with the same input and output length. For convenience we refer to this random function as Fun. Using algorithms $\mathcal{E}$ and $\mathcal{V}$ shown in Figure 6, we can reduce the distinguishability of Equation (2) to the distinguishability of the pair of oracles $(\mathcal{E}_{F_k}, \mathcal{V}_{F_k})$ and $(\mathcal{E}_{\mathrm{Fun}}, \mathcal{V}_{\mathrm{Fun}})$. We base this reduction on adversary $B_{A,\Psi}$ given in Figure 4.

In our description of $B_{A,\Psi}$, we use the parameter Ψ to indicate the hardwiring of B's oracle query to $\mathcal{E}$ (which is dependent on StateTable(Ψ)). $B_{A,\Psi}$ uses $\mathcal{E}$'s response to construct the obfuscated code, which is given to A. Using A, $B_{A,\Psi}$ simulates A's query-response interaction with the oracle. The distinguishing bit b returned by $B_{A,\Psi}$ is the same bit returned by A. Therefore Equation (6) reduces to Equation (7). If we replace every oracle call to $\mathcal{E}$ and $\mathcal{V}$ with multiple calls to either F_K or Fun then we can reduce Equation (7) even further. We denote this simulation by $B_{A,\Psi}{}'$ to distinguish itself from $B_{A,\Psi}$. Therefore Equation (7) reduces to Equation (8). But this last equation is just the pseudorandom distinguishability of F_K given auxiliary input z. Using our assumption that non-uniformly strong one-way functions exist, we can use the Goldreich et al. construction in [9] to generate a pseudorandom function that is secure against non-uniform PPT adversaries (denoted as prf-nu). If the adversary A makes no more than q_v distinct[3] **State_Update** queries, then the total number of queries made to F_K or Fun by $B_{A,\Psi}{}'$ is no more than $(q_v + 2)|E(\Psi)| + 1$.

[3] Each q_v represents a complete chain of **State_Update** queries (i.e., the user has submitted the entire encrypted table with *Auth* tag).

Setup of $B_{A,\Psi}$:

INPUT: $1^k, z$

GENERATE STATE TABLE:
$(|m|, |Table|, \mathbb{T}_{state}) \leftarrow \text{STATETABLE}(\Psi)$

ENCRYPT STATE TABLE ENTRIES:
Query oracle $\mathcal{E}$ with $\mathbb{T}_{state}$
$(\mathbb{T}_C, Auth) \Leftarrow \mathcal{E}(\mathbb{T}_{state})$

$A \Leftarrow \mathcal{O}(|Table|, \mathbb{T}_C, Auth), z$

Simulation of Oracle $\mathcal{R}$:

INPUT: $1^k, |m|, |Table|, \mathbb{T}_{state}, \mathbb{T}_C, Auth$

INITIALIZATION:
$current\_state \leftarrow 0$
$acpt \leftarrow \perp$
$temp_\alpha \leftarrow \perp$
$temp_{cs} \leftarrow \perp$
$flag_{auth} \leftarrow \perp$
$C \leftarrow \perp$
$s \leftarrow \perp$
$State \leftarrow$ **Transition_Query**

Case$(State)$
Transition_Query:
When A makes a query α **do**
$temp_\alpha \leftarrow \alpha$
$flag_{auth} \leftarrow false$
$C \leftarrow \perp$
$s \leftarrow 0$
$State \leftarrow$ **State_Update**

State_Update:
When A makes a query $\mathbb{T}'_C[s]$ **or**
$\mathbb{T}'_C[s] \| Auth'$ **do**

STATE AUTHENTICATION:
$C \leftarrow C \| \mathbb{T}'_C[s]$
if $\mathbb{T}'_C[s] \neq \mathbb{T}_C[s]$ **or** $(s = |Table| - 1$ **and**
$Auth' \neq Auth)$ **then**
$flag_{auth} \leftarrow true$
if $s = |Table| - 1$ **and** $flag_{auth} = true$ **then**
Query oracle $\mathcal{V}$ with $(C, Auth')$
if $0 \Leftarrow \mathcal{V}(C, Auth')$ **then**
$A \Leftarrow auth\_fail$
$State \leftarrow$ **Transition_Query**

COMPARE TABLE ENTRIES:
$M'_s \leftarrow \mathbb{T}'_C[s] \oplus (\mathbb{T}_C[s] \oplus \mathbb{T}_{state}[s])$
$s_\alpha \| s_{state} \| s_{\delta(state,\alpha)} \| s_{acpt} \leftarrow M'_{s[k-1:k-|m|]}$
if $s_{state} = current\_state$ **and** $s_\alpha = temp_\alpha$ **then**
$temp_{cs} \leftarrow s_{\delta(state,\alpha)}$
$acpt \leftarrow s_{acpt}$

UPDATE ORACLE STATE:
if $s = |Table| - 1$ **then**
$current\_state \leftarrow temp_{cs}$
$A \Leftarrow acpt$
$State \leftarrow$ **Transition_Query**
$s \leftarrow s + 1$

Fig. 4. Adversary $B_{A,\Psi}$.

Therefore Equation (6) reduces to Equation (10), which is negligible following our assumption.

$$\left|\Pr[A^{\mathcal{R}_{F_K}}(\mathcal{O}^{\mathcal{R}_{F_K}}(\Psi), z) = 1] - \Pr[A^{\mathcal{R}_{\text{Fun}}}(\mathcal{O}^{\mathcal{R}_{\text{Fun}}}(\Psi), z) = 1]\right| \tag{6}$$

$$= \left|\Pr[B_{A,\Psi}^{\mathcal{E}_{F_K}, \mathcal{V}_{F_K}}(z) = 1] - \Pr[B_{A,\Psi}^{\mathcal{E}_{\text{Fun}}, \mathcal{V}_{\text{Fun}}}(z) = 1]\right| \tag{7}$$

$$= \left|\Pr[B_{A,\Psi}^{F_K\prime}(z) = 1] - \Pr[B_{A,\Psi}^{\text{Fun}\prime}(z) = 1]\right| \tag{8}$$

$$= \mathbf{Adv}^{\text{prf}}_{F_K, B_{A,\Psi}\prime}(k, z) \tag{9}$$

$$\leq \mathbf{Adv}^{\text{prf-nu}}_{F_K}(k, (q_v + 2)|E(\Psi)| + 1). \tag{10}$$

For Equation (3), we would like to perform a similar reduction as performed for Equation (8) except, instead of measuring the pseudorandomness of F_K, we would like to measure the unforgeability provided by the verifier $\mathcal{V}$. To do this, we introduce the oracle $\mathcal{R}^*_{\text{Fun}}$. Internally, the oracle $\mathcal{R}^*_{\text{Fun}}$ looks identical to $\mathcal{R}_{\text{Fun}}$ except during the state authentication process. Instead of computing

a partial authentication tag for each **State_Update** query, as is done in Figure 3, it collectively gathers all of the ciphertext queries and final authentication tag and submits them to a verifier $\mathcal{V}^*$. To do this, $\mathcal{R}^*_{\text{Fun}}$ stores the values $(\mathbb{T}_C, \mathit{Auth})$ returned by the initial $\mathbf{Setup}(M_\Psi, k)$ algorithm. During the **State_Update** phase, the oracle checks to see if the table entries queried by the user are the same entries as those in $\mathbb{T}_C$. If any of the table entries are incorrect, including the final authentication tag, or if they are queried in a different order, the oracle $\mathcal{R}^*_{\text{Fun}}$ returns *auth_fail*. Reusing $B_{A,\Psi}$ we can reduce Equation (3) to inequality (11) by simulating the distinguishability with oracles $(\mathcal{E}_{\text{Fun}}, \mathcal{V}_{\text{Fun}})$ and $(\mathcal{E}_{\text{Fun}}, \mathcal{V}^*)$. We call this advantage IND-VERF, since it measures the indistinguishability between the two verifiers. Therefore

$$\begin{aligned}
&\left|\Pr[A^{\mathcal{R}_{\text{Fun}}}(\mathcal{O}^{\mathcal{R}_{\text{Fun}}}(\Psi), z) = 1] - \Pr[A^{\mathcal{R}^*_{\text{Fun}}}(\mathcal{O}^{\mathcal{R}^*_{\text{Fun}}}(\Psi), z) = 1]\right| \\
&= \left|\Pr[B^{\mathcal{E}_{\text{Fun}}, \mathcal{V}_{\text{Fun}}}_{A,\Psi}(z) = 1] - \Pr[B^{\mathcal{E}_{\text{Fun}}, \mathcal{V}^*}_{A,\Psi}(z) = 1]\right| \\
&= \mathbf{Adv}^{\text{ind-verf}}_{\mathcal{SE}_{\text{Fun}}, B_{A,\Psi}}(k, q_e, q_v, \eta_e, \eta_v, z). \qquad (11)
\end{aligned}$$

with $q_e = 1$ denoting the number of encryption queries and $\eta_e = \eta_v - 1 = |E(\Psi)|$ the maximum number of k-bit blocks each encryption or verification query may have. We claim this advantage is bounded above by the INT-CTXT-m security of $\mathcal{SE}_{\text{Fun}}$. See Appendix A.1 for more details on the security definition of INT-CTXT-m.

Claim. $\mathbf{Adv}^{\text{ind-verf}}_{\mathcal{SE}_{\text{Fun}}, B_{A,\Psi}}(k, q_e, q_v, \eta_e, \eta_v, z) \leq \mathbf{Adv}^{\text{int-ctxt-m}}_{\mathcal{SE}_{\text{Fun}}, (B_{A,\Psi})_{\text{ctxt}}}(k, q_e, q_v, \eta_e, \eta_v, z)$

See the extended abstract [1] for a proof of this claim. Now that we have bounded Equation (3) by the INT-CTXT-m security of $\mathcal{SE}_{\text{Fun}}$ we are now ready to move onto Equation (4).

In Equation (4) we measure the chosen plaintext distinguishability between encrypting with either $\mathcal{E}_{\text{Fun}}$ or $\mathcal{E}_{\text{Rand}}$, where $\mathcal{E}_{\text{Rand}}(M)$ is a random string of length $|M|$. The oracles $\mathcal{R}^*_{\text{Rand}}$ and $\mathcal{R}^*_{\text{Fun}}$ are identical except for their calls to $\mathcal{E}_{\text{Fun}}$ or $\mathcal{E}_{\text{Rand}}$. As before we will use the $*$ in $\mathcal{R}^*_{\text{Rand}}$ to denote that verifier $\mathcal{V}^*$ is used. We define $B^*_{A,\Psi}$ to be the algorithm $B_{A,\Psi}$ that uses $\mathcal{V}^*$ as its verifier (which can be easily *simulated* given the output of $\mathcal{E}$). Therefore Equation (4) reduces to inequality (12)

$$\begin{aligned}
&\left|\Pr[A^{\mathcal{R}^*_{\text{Fun}}}(\mathcal{O}^{\mathcal{R}^*_{\text{Fun}}}(\Psi), z) = 1] - \Pr[A^{\mathcal{R}^*_{\text{Rand}}}(\mathcal{O}^{\mathcal{R}^*_{\text{Rand}}}(\Psi), z) = 1]\right| \\
&= \left|\Pr[B^{*\,\mathcal{E}_{\text{Fun}}}_{A,\Psi}(z) = 1] - \Pr[B^{*\,\mathcal{E}_{\text{Rand}}}_{A,\Psi}(z) = 1]\right| \\
&= \mathbf{Adv}^{\text{ind\$-cpa}}_{\mathcal{SE}_{\text{Fun}}, B^*_{A,\Psi}}(k, q_e, \eta_e, z). \qquad (12)
\end{aligned}$$

with $q_e = 1$ and $\eta_e = |E(\Psi)|$.

In the final Equation (5), we introduce the simulator S, which as you recall has only black box access to Ψ. In order for S to properly simulate A's view, it needs to know the number of edges $|E(\Psi)|$. This can be easily extracted knowing just the size of M_Ψ based on our encoding. Given the number of edges $|E(\Psi)|$, S can easily simulate A's view of the obfuscated code by giving A a copy of $\mathcal{O}(|E(\Psi)|, \mathbb{T}_C, \mathit{Auth})$, where $\mathbb{T}_C$ is a random table of the appropriate size (dependent on $|E(\Psi)|$ and k) and Auth a k-bit random string. Using its oracle access to Ψ, S can simulate A's interaction with $\mathcal{R}^*_{\text{Rand}}$ using the values $|E(\Psi)|, \mathbb{T}_C$, and Auth. Therefore, the entire simulation, which we denote by S_A, consists of passing A the obfuscated code $\mathcal{O}(|E(\Psi)|, \mathbb{T}_C, \mathit{Auth})$ and simulating the interaction between $\mathcal{R}^*_{\text{Rand}}$ and A using oracle Ψ. The full description of simulator S_A is given in Figure 5.

Setup of S_A:

INPUT: $1^k, 1^{|E(\Psi)|}, z$

GENERATE STATE TABLE:
  for $s \leftarrow 0$ **to** $|E(\Psi)| - 1$ **do**
    $\mathbb{T}_{state}[s] \leftarrow 0^k$

ENCRYPT STATE TABLE ENTRIES:
  Query $\mathcal{E}_{\text{Rand}}$ with $\mathbb{T}_{state}$
  $(\mathbb{T}_C, Auth) \Leftarrow \mathcal{E}_{\text{Rand}}(\mathbb{T}_{state})$

  $A \Leftarrow \mathcal{O}(|E(\Psi)|, \mathbb{T}_C, Auth), z$

Simulation of Oracle $\mathcal{R}^*_{\text{Rand}}$:

INPUT: $|E(\Psi)|, \mathbb{T}_C, Auth$

INITIALIZATION:
  $acpt \leftarrow \perp$
  $temp_\alpha \leftarrow \perp$
  $flag_{auth} \leftarrow \perp$
  $C \leftarrow \perp$
  $s \leftarrow \perp$
  $State \leftarrow$ **Transition_Query**

Case($State$)
Transition_Query:
  When A makes a query α **do**
    $temp_\alpha \leftarrow \alpha$
    $flag_{auth} \leftarrow false$
    $C \leftarrow \perp$
    $s \leftarrow 0$
    $State \leftarrow$ **State_Update**

State_Update:
  When A makes a query $\mathbb{T}'_C[s]$ **or**
    $\mathbb{T}'_C[s] \| Auth'$ **do**

  STATE AUTHENTICATION:
    $C \leftarrow C \| \mathbb{T}'_C[s]$
    if $\mathbb{T}'_C[s] \neq \mathbb{T}_C[s]$ **or** $(s = |E(\Psi)| - 1$ **and**
      $Auth' \neq Auth)$ **then**
      $flag_{auth} \leftarrow true$
    if $s = |E(\Psi)| - 1$ **and** $flag_{auth} = true$ **then**
      Query $\mathcal{V}^*$ with $(C, Auth')$
      if $0 \Leftarrow \mathcal{V}^*(C, Auth')$ **then**
        $A \Leftarrow auth\_fail$
        $State \leftarrow$ **Transition_Query**

  QUERY DFA ORACLE:
    if $s = |E(\Psi)| - 1$ **then**
      Query oracle Ψ with $temp_\alpha$
      $acpt \leftarrow \Psi(temp_\alpha)$

  UPDATE ORACLE STATE:
    if $s = |E(\Psi)| - 1$ **then**
      $A \Leftarrow acpt$
      $State \leftarrow$ **Transition_Query**
    $s \leftarrow s + 1$

Fig. 5. Simulator S_A.

To help with the analysis, we model adversary $A^{\mathcal{R}^*_{\text{Rand}}}(\mathcal{O}^{\mathcal{R}^*_{\text{Rand}}}(\Psi), z)$ as we did in Equation (4) by replacing it with $B^{*\mathcal{E}_{\text{Rand}}}_{A,\Psi}(z)$. From this we have $\Pr[A^{\mathcal{R}^*_{\text{Rand}}}(\mathcal{O}^{\mathcal{R}^*_{\text{Rand}}}(\Psi), z) = 1] = \Pr[B^{*\mathcal{E}_{\text{Rand}}}_{A,\Psi}(z) = 1]$. Notice that during the **State_Update** phase of $B^*_{A,\Psi}$, in order for the final query to reach UPDATE ORACLE STATE and return an output other than $auth\_fail$, $\mathcal{R}^*_{\text{Rand}}$ must pass the verifier $\mathcal{V}^*$. This implies that the adversary submits the table $\mathbb{T}_C$ free of modifications. Hence the operations under COMPARE TABLE ENTRIES may be completely replaced with a simulated oracle call to the DFA in much the same way simulator S_A does. Replacing this code, we obtain a new ${B^*_{A,\Psi}}'$ which is functionally equivalent to $B^*_{A,\Psi}$. Since the variables $current\_state$ and $temp_{cs}$ are no longer needed as they are used in the simulation of oracle Ψ, we can remove them. Finally, observe that an oracle call to $\mathcal{E}_{\text{Rand}}$ in ENCRYPT STATE TABLE ENTRIES returns random strings regardless of the particular input. Therefore encrypting with the real state table $\mathbb{T}_{state}$ or one containing all zeroes provides a random output that is of the same size. Hence it follows ${B^*_{A,\Psi}}'$ and S_A have a distinguishability of 0.

Using the bounds derived in Appendix A with $q_e = 1$ and $\eta_e = \eta_v - 1 = |E(\Psi)|$, we have the following result

$$\begin{aligned}
&\left|\Pr[A^{\mathcal{R}_{F_K}}(\mathcal{O}^{\mathcal{R}_{F_K}}(\Psi), z) = 1] - \Pr[S^{\Psi}(1^{|E(\Psi)|}, z) = 1]\right| \qquad (13)\\
&\leq \mathbf{Adv}^{\text{prf-nu}}_{F_K}(k, (q_v + 2)|E(\Psi)| + 1)\\
&\qquad + \mathbf{Adv}^{\text{int-ctxt-m}}_{\mathcal{SE}_{\text{Fun}}, (B_{A,\Psi})_{\text{ctxt}}}(k, q_e, q_v, \eta_e, \eta_v, z) + \mathbf{Adv}^{\text{ind\$-cpa}}_{\mathcal{SE}_{\text{Fun}}, B^*_{A,\Psi}}(k, q_e, \eta_e, z)\\
&\leq \mathbf{Adv}^{\text{prf-nu}}_{F_K}(k, (q_v + 2)|E(\Psi)| + 1)\\
&\qquad + q_v(4|E(\Psi)|^2 + |E(\Psi)|)2^{-k} + \frac{1}{2}(3|E(\Psi)|^2 + |E(\Psi)|)2^{-k}.
\end{aligned}$$

□

The amount of persistent state needed to obfuscate the DFA in the above Proposition is in fact quite small. In the next Proposition we show that we need at most $O(k)$-bits. This is especially ideal if the oracle is implemented on a computationally limited device with a minimal amount of tamper protection.

Corollary 1 *If non-uniformly strong one-way functions exist, then non-resettable DFAs are obfuscatable with respect to oracle machines with small internal state.*

Proof: In Proposition 1 we used the Goldreich et al. construction in [9] to generate a pseudorandom function that is secure against non-uniform PPT adversaries. The key generated for this construction is the same size as the security parameter k. But this implies that the size of the oracle's persistent internal state is no more than $O(\log |State(\Psi)| + \log |\Sigma| + k) = O(k)$, following our definition of $\mathcal{F}_k$. □

References

1. W. E. Anderson "On the Secure Obfuscation of Deterministic Finite Automata: Extended Abstract", *Crypto ePrint Archive*, 2008/184.
2. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, K. Yang "On the (Im)possibility of Obfuscating Programs", in *Advances in Cryptology - CRYPTO 2001*. LNCS, vol. 2139 (Springer, Berlin, 2001), pp. 1-18.
3. M. Bellare, O. Goldreich, A. Mityagin "The Power of Verification Queries in Message Authentication and Authenticated Encryption", *Crypto ePrint Archive*, 2004/309.
4. M. Bellare, P. Rogaway "The Security of Triple Encryption and a Framework for Code-Bases Game-Playing Proofs", in *Advances in Cryptology - EUROCRYPT 2006*, LNCS, vol. 4004 (Springer, Berlin, 2006), pp. 409-426.
5. R. Best "Microprocessor for Executing Encrypted Programs", US Patent 4,168,396. Issued September 1979.
6. R. Canetti "Towards Realizing Random Oracles: Hash Functions that Hide all Partial Information", in *Advances in Cryptology - CRYPTO 1997*, LNCS, vol. 1294 (Springer, Berlin, 1997), pp. 455-469.
7. R. Canetti, D. Micciancio, O. Reingold "Perfectly One-way Probabilistic Hash Function", in *Proceedings 30th ACM - STOC 1998*, pp. 131-140.
8. O. Goldreich "Towards a Theory of Software Protection and Simulation by Oblivious RAMS", in *Proceedings 19th ACM - STOC 1987*, pp. 182-194.
9. O. Goldreich, S. Goldwasser, S. Micali, "How to Construct Random Functions", in *J. of the ACM*, vol. 33, no. 4, 1986, pp. 792-807.
10. O. Goldreich, R. Ostrovsky "Software Protection and Simulation on Oblivious RAMs", in *J. of the ACM* vol. 43, no. 3, 1996, pp. 431-473.

11. S. Goldwasser, S. Micali, C. Rackoff "The Knowledge Complexity of Interactive Proof Systems", in *SIAM Journal on Computing*, vol. 18, no. 1, 1989, pp. 186-208.
12. S. Goldwasser, Y. Tauman Kalai "On the Impossibility of Obfuscation with Auxiliary Input", in *Proceedings 46th IEEE - FOCS 2005*, pp. 553-562.
13. S.T. Kent "Protecting Externally Supplied Software in Small Computers", Ph.D. Thesis, MIT/LCS/TR-255 1980.
14. B. Lynn, M. Prabhakaran, A. Sahai "Positive Results and Techniques for Obfuscation", in *Advances in Cryptology - EUROCRYPT 2004*, LNCS, vol. 3027, (Springer, Berlin, 2004), pp. 20-39.
15. H. Wee "On Obfuscating Point Functions", in *Proceedings 37th ACM - STOC 2005*, pp. 523-532.

A Supplementary Proofs

In this appendix we review the security definitions of INT-CTXT and IND$-CPA and prove the bounds used in inequality 13. All of the results proven below are based on the authenticated encryption scheme shown in Figure 6.

Algorithm $\mathcal{E}_\rho(M)$	Algorithm $\mathcal{V}_\rho(C'\|Auth')$
$C \leftarrow \perp$	${C_0}'\| \ldots \|{C_{t-1}}' \leftarrow C'$
$M_0\| \ldots \|M_{t-1} \leftarrow M$, $\|M_i\| = k$	$X_1^{-1} \leftarrow 1^k$
$X_1^{-1} \leftarrow 1^k$	$Auth \leftarrow \rho(X_1^{-1})$
$Auth \leftarrow \rho(X_1^{-1})$	**for** $s \leftarrow 0$ **to** $t-1$ **do**
for $s \leftarrow 0$ **to** $t-1$ **do**	$X_1^{s\prime} \leftarrow Auth \oplus {C_s}'$
$X_0^s \leftarrow s\|0$	$Auth \leftarrow \rho(X_1^s)$
$Y \leftarrow \rho(X_0^s)$	**if** $Auth = Auth'$ **and**
$C_s \leftarrow Y \oplus M_s$	$\mathsf{size}_C = \|C'\|$ **then**
$C \leftarrow C\|C_s$	**Return** 1, else **Return** 0.
$X_1^s \leftarrow Auth \oplus C_s$	
$Auth \leftarrow \rho(X_1^s)$	
$\mathsf{size}_C \leftarrow \|C\|$	
Return $(C\|Auth)$.	

Fig. 6. Encryption and Verification Schemes.

A.1 Integrity Awareness

In Proposition 1 we showed that the distinguishing advantage between the verifiers $\mathcal{V}_{\text{Fun}}$ and $\mathcal{V}^*$ (with the adversary also having access to $\mathcal{E}_{\text{Fun}}$) is bounded above by the strong unforgeability of the ciphertexts. We state the security definition formally below.

Definition 2. (Integrity Awareness w.r.t. Auxiliary Input): *Let $\mathcal{SE}_{\text{Fun}}$ be the symmetric encryption scheme in Figure 6 using random functions and A_{ctxt} a PPT adversary with access to two oracles, $\mathcal{E}_{\text{Fun}}$ and $\mathcal{V}_{\text{Fun}}$. Consider the following experiment with $k \in \mathbb{N}$ and $z \in \{0,1\}^{q(k)}$ for some polynomial q*

Experiment $\mathbf{Exp}^{\text{int-ctxt-m}}_{\mathcal{SE}_{\text{Fun}},A_{\text{ctxt}}}(k,z)$
$\text{Fun} \xleftarrow{\$} \mathsf{Fun}(k)$
If $A^{\mathcal{E}_{\text{Fun}},\mathcal{V}_{\text{Fun}}}_{\text{ctxt}}(k,z)$ makes a query C to the oracle $\mathcal{V}_{\text{Fun}}$ such that
- $\mathcal{V}_{\text{Fun}}(C) = 1$
- C was never a response to $\mathcal{E}_{\text{Fun}}$
then Return 1 else Return 0.

We denote the winning probability in adversary A_{ctxt} breaking INT-CTXT-m as

$$\mathbf{Adv}^{\text{int-ctxt-m}}_{\mathcal{SE}_{\text{Fun}},A_{\text{ctxt}}}(k,z) := \Pr[\mathbf{Exp}^{\text{int-ctxt-m}}_{\mathcal{SE}_{\text{Fun}},A_{\text{ctxt}}}(k,z) = 1]$$

The INT-CTXT-m advantage over all PPT adversaries A_{ctxt} is defined as the maximum

$$\mathbf{Adv}^{\text{int-ctxt-m}}_{\mathcal{SE}_{\text{Fun}}}(k,q_e,q_v,\eta_e,\eta_v,z) := \max_{A_{\text{ctxt}}}\{\mathbf{Adv}^{\text{int-ctxt-m}}_{\mathcal{SE}_{\text{Fun}},A_{\text{ctxt}}}(k,z)\}$$

where q_e and q_v denote the maximum number of oracle calls to $\mathcal{E}_{\text{Fun}}$ and $\mathcal{V}_{\text{Fun}}$, while η_e and η_v denote the maximum number of k-bit blocks per encryption and verification query. The scheme $\mathcal{SE}_{\text{Fun}}$ is said to be INT-CTXT-m secure w.r.t. auxiliary input if the advantage $\mathbf{Adv}^{\text{int-ctxt-m}}_{\mathcal{SE}_{\text{Fun}}}$ is negligible over all PPT adversaries (with time-complexity polynomial bounded in k) given arbitrary auxiliary input.

In the special case where we allow only a single verification query $q_v = 1$, we define the advantage as INT-CTXT-1. It was shown by Bellare et al. in [3] that if an encryption scheme $\mathcal{SE}$ is INT-CTXT-1 secure (without an auxiliary input), then it is also INT-CTXT-m secure. Adding auxiliary inputs is a trivial modification to the original proof. Since we will be using this result to simplify our analysis, we state it in the following lemma.

Lemma 1. (INT-CTXT-1 ⇒ INT-CTXT-M [3]) *Let $\mathcal{SE}$ be any symmetric encryption scheme and z any polynomial bounded string in k with $k \geq 1$. Then*

$$\mathbf{Adv}^{\text{int-ctxt-m}}_{\mathcal{SE}}(k,q_e,q_v,\eta_e,\eta_v,z) \leq q_v \cdot \mathbf{Adv}^{\text{int-ctxt-1}}_{\mathcal{SE}}(k,q_e,\eta_e,\eta_v,z)$$

In the following Proposition we prove the scheme in Figure 6 is INT-CTXT-m secure when $q_e = 1$. This result is used to help facilitate the proof in Proposition 1.

Proposition 2 *Let $\mathcal{SE}_{\text{Fun}}$ be the scheme given in Figure 6. Let z be any polynomial bounded string in k with $q_e = 1$, $\eta_v = \eta_e + 1$, and $q_v, k \geq 1$. Then*

$$\mathbf{Adv}^{\text{int-ctxt-m}}_{\mathcal{SE}_{\text{Fun}}}(k,q_e,q_v,\eta_e,\eta_v,z) \leq q_v(4\eta_e^2 + \eta_e)2^{-k}$$

Proof: To prove the above inequality holds, we will use the game-playing techniques introduced by Bellare and Rogaway in [4]. Our goal is to incrementally construct a chain of games using simple transformation techniques so that the terminal game is bounded above by a negligible factor. To simplify our analysis we use the result of Lemma 1 and derive an upperbound for INT-CTXT-1. Once we have found a bound for INT-CTXT-1, the more general INT-CTXT-m bound will follow. For the sake of this proof, we will also assume that our adversary A is computationally

unbounded and therefore deterministic (since it may deterministically choose its queries to maximize its advantage). The only restrictions we place on A is the number of queries it can make.

We begin our analysis by giving a description of game $\mathsf{G1}$ shown in Figure 8. Notice that the scheme $\mathcal{SE}_{\text{Fun}}$ is not stateful and therefore not IND-CPA secure. Having IND-CPA security is not essential to proving the claim since $q_e = 1$. Also observe that we removed the checking of size_C in game $\mathsf{G1}$ since the adversary does not gain an advantage by submitting a ciphertext authentication pair of a different length. We will instead assume without loss of generality that the pair submitted for verification is the same size as the pair returned by the encryption query. Let ρ be a randomly (independent of z) chosen function from the set $\mathsf{Fun}(k)$. Observe that game $\mathsf{G1}$ has only two queries in its description: an encryption query and a verification query. The single encryption query ($q_e = 1$) simulates obfuscating a single DFA while the verification query ($q_v = 1$) is the result of restricting our analysis to INT-CTXT-1. Based on the description of game $\mathsf{G1}$ it follows that

$$\mathbf{Adv}^{\text{int-ctxt-1}}_{\mathcal{SE}_{\text{Fun}}}(k, q_e, \eta_e, \eta_v, z) = \Pr[\text{Game } \mathsf{G1} \text{ sets } \mathit{bad}]$$

with $q_e = 1$ and $\eta_e = \eta_v - 1 = t$.

To transform game $\mathsf{G1} \to \mathsf{G2}$, we add additional settings of *bad* in lines 208, 214, and 224. We also observe that during the second query, the *Auth* value after the first index i where $C_i' \neq C_i$ is just $\rho(X_1^{i-1})$. Therefore, the modifications made in lines 219 through 225 are a direct result of this observation. Since the functionality of game $\mathsf{G1}$ and $\mathsf{G2}$ are equivalent with the exception of additional settings of *bad* it follows that $\Pr[\text{Game } \mathsf{G1}] \leq \Pr[\text{Game } \mathsf{G2}]$.

To go from game $\mathsf{G2} \to \mathsf{G3}$, we unroll the **for** loops in line 205 and 221 and postpone the recordings of the variable X_1^s in $\text{Dom}(\rho)$. We also swap the assignment of the variable $X_1^s \leftarrow \mathit{Auth} \oplus C_s$ with a random sampling $X_1^s \xleftarrow{\$} \{0,1\}^k$, since the *Auth* variable used in the assignment of X_1^s is randomly sampled during $s-1$. Finally, the assignments occurring after the setting of $\mathit{bad} \leftarrow \mathit{true}$ are removed. Therefore, the changes made from game $\mathsf{G2}$ to $\mathsf{G3}$ are conservative (i.e. $\Pr[\text{Game } \mathsf{G2}] = \Pr[\text{Game } \mathsf{G3}]$).

For the final game $\mathsf{G3} \to \mathsf{G4}$ we begin by first swapping the random-assignment in line 305 with line 308 by replacing $Y \xleftarrow{\$} \{0,1\}^k$ and $C_s \leftarrow Y \oplus M_s$ with $C_s \xleftarrow{\$} \{0,1\}^k$ and $Y \leftarrow C_s \oplus M_s$. Since the variable Y is no longer used, we may eliminate it from the game. Similarly, since the values recorded for $\rho(X_1^s)$ and $\rho(X_0^s)$ are never reused, they may be arbitrarily renamed as $\mathsf{defined}$. The only prerecorded variable that is reused is X_1^i on line 413. Given the above swapping it is easy to see that both C and *Auth* are random. Using the derandomization technique[4] we may replace them with constants $\mathtt{C}\|\mathtt{Auth}$. Since adversary A is deterministic, there exist queries $\mathtt{M}_0\| \ldots \|\mathtt{M}_{t-1}$ and $\mathtt{C}'\|\mathtt{Auth}'$ corresponding to output $\mathtt{C}\|\mathtt{Auth}$. By hardwiring these query-responses into game $\mathsf{G4}$, we may bound the probability of setting *bad* as the maximum over all the possible query-responses (thus removing the adaptivity of the adversary). It is not difficult to see that this maximum occurs when $t = \eta_e$, and the adversary submits a $t+1$-block authentication query with the first ciphertext block changed. Since there are $t+1$ non-random variables $X_0^{s=0,\ldots,t-1}, X_1^{-1}$ that do not collide with one another and $2t-1$ independent random variables $X_1^{s=0,\ldots,t-1}, X_1^{s=1,\ldots,t-1'}$ with a single dependent random variable $X_1^{0'} = X_1^0 \oplus \delta$ some fixed $\delta \neq 0$ recorded in $\text{Dom}(\rho)$, it follows that the

[4] Derandomization Technique: If a game G chooses a variable $X \xleftarrow{\$} \mathcal{X}$ and never redefines it, we may derandomize the variable by choosing a constant $\mathtt{X}$ to replace it. Given any adversary A, it follows that $\Pr[\text{Game } \mathsf{G}_A \text{ sets } \mathit{bad}] \leq \max_{\mathtt{X}} \Pr[\text{Game } \mathsf{G}_A^{\mathtt{X}} \text{ sets } \mathit{bad}]$.

Game G1

100 **On first query** $M_0\|\ldots\|M_{t-1}$
101 $C \leftarrow \perp$
102 $X_1^{-1} \leftarrow 1^k$
103 $Auth \xleftarrow{\$} \{0,1\}^k$
104 $\rho(X_1^{-1}) \leftarrow Auth$
105 **for** $s \leftarrow 0$ **to** $t-1$ **do**
106 $X_0^s \leftarrow s\|0$
107 $Y \xleftarrow{\$} \{0,1\}^k$
108 **if** $X_0^s \in \mathrm{Dom}(\rho)$ **then** $Y \leftarrow \rho(X_0)$
109 $\rho(X_0^s) \leftarrow Y$
110 $C_s \leftarrow Y \oplus M_s$
111 $C \leftarrow C\|C_s$
112 $X_1^s \leftarrow Auth \oplus C_s$
113 $Auth \xleftarrow{\$} \{0,1\}^k$
114 **if** $X_1^s \in \mathrm{Dom}(\rho)$ **then** $Auth \leftarrow \rho(X_1^s)$
115 $\rho(X_1^s) \leftarrow Auth$
116 **Return** $C\|Auth$

117 **On second query** $C'\|Auth'$
118 $C_0'\|\ldots\|C_{t-1}' \leftarrow C'$
119 $Auth \leftarrow \rho(X_1^{-1})$
120 **for** $s \leftarrow 0$ **to** $t-1$ **do**
121 $X_1^{s\prime} \leftarrow Auth \oplus C_s'$
122 $Auth \xleftarrow{\$} \{0,1\}^k$
123 **if** $X_1^{s\prime} \in \mathrm{Dom}(\rho)$ **then** $Auth \leftarrow \rho(X_1^{s\prime})$
124 $\rho(X_1^{s\prime}) \leftarrow Auth$
125 $b \leftarrow 0$
126 **if** $Auth = Auth'$ **then** $bad \leftarrow true$, $b \leftarrow 1$
127 **Return** b

Game G2

200 **On first query** $M_0\|\ldots\|M_{t-1}$
201 $C \leftarrow \perp$
202 $X_1^{-1} \leftarrow 1^k$
203 $Auth \xleftarrow{\$} \{0,1\}^k$
204 $\rho(X_1^{-1}) \leftarrow Auth$
205 **for** $s \leftarrow 0$ **to** $t-1$ **do**
206 $X_0^s \leftarrow s\|0$
207 $Y \xleftarrow{\$} \{0,1\}^k$
208 **if** $X_0^s \in \mathrm{Dom}(\rho)$ **then** $bad \leftarrow true$, $Y \leftarrow \rho(X_0^s)$
209 $\rho(X_0^s) \leftarrow Y$
210 $C_s \leftarrow Y \oplus M_s$
211 $C \leftarrow C\|C_s$
212 $X_1^s \leftarrow Auth \oplus C_s$
213 $Auth \xleftarrow{\$} \{0,1\}^k$
214 **if** $X_1^s \in \mathrm{Dom}(\rho)$ **then** $bad \leftarrow true$,
$Auth \leftarrow \rho(X_1^s)$
215 $\rho(X_1^s) \leftarrow Auth$
216 **Return** $C\|Auth$

217 **On second query** $C'\|Auth'$
218 $C_0\|\ldots\|C_{i-1}\|C_i'\|\ldots\|C_{t-1}' \leftarrow C'$
219 $i \leftarrow \min\{s \mid C_s' \neq C_s\}$
220 $Auth \leftarrow \rho(X_1^{i-1})$
221 **for** $s \leftarrow i$ **to** $t-1$ **do**
222 $X_1^{s\prime} \leftarrow Auth \oplus C_s'$
223 $Auth \xleftarrow{\$} \{0,1\}^k$
224 **if** $X_1^{s\prime} \in \mathrm{Dom}(\rho)$ **then** $bad \leftarrow true$,
$Auth \leftarrow \rho(X_1^{s\prime})$
225 $\rho(X_1^{s\prime}) \leftarrow Auth$
226 $b \leftarrow 0$
227 **if** $Auth = Auth'$ **then** $bad \leftarrow true$, $b \leftarrow 1$
228 **Return** b

Fig. 7. INT-CTXT-1 Games G1-G2.

setting of *bad* based on these variables is

$$\Pr[\text{Variables in } \mathrm{Dom}(\rho) \text{ set } bad] \leq \left\{ \binom{3t+1}{2} - \binom{t+1}{2} - 1 \right\} 2^{-k}$$

which holds for any computationally unbounded adversary. Therefore, given $q_e = 1$, $\eta_v = \eta_e + 1$, and $\Pr[Auth \text{ sets } bad \text{ in line } 423] = 2^{-k}$ we have

$$\begin{aligned}
\mathbf{Adv}^{\text{int-ctxt-1}}_{\mathcal{SE}_{\mathrm{Fun}}}(k, q_e, \eta_e, \eta_v, z) &\leq \Pr[\text{Game G4 sets } bad] \\
&\leq \Pr[\text{Variables in } \mathrm{Dom}(\rho) \text{ set } bad] \\
&\qquad + \Pr[Auth \text{ sets } bad \text{ in line } 423] \\
&\leq \left\{ \binom{3\eta_e+1}{2} - \binom{\eta_e+1}{2} \right\} 2^{-k} \\
&= (4\eta_e^2 + \eta_e) 2^{-k}.
\end{aligned}$$

□

Game G3

300 **On first query** $M_0 \| \ldots \| M_{t-1}$
301 $C \leftarrow \perp$
302 $X_1^{-1} \leftarrow 1^k$
303 **for** $s \leftarrow 0$ **to** $t-1$ **do**
304 $X_0^s \leftarrow s \| 0$
305 $Y \xleftarrow{\$} \{0,1\}^k$
306 **if** $X_0^s \in \mathrm{Dom}(\rho)$ **then** $bad \leftarrow true$
307 $\rho(X_0^s) \leftarrow Y$
308 $C_s \leftarrow Y \oplus M_s$
309 $C \leftarrow C \| C_s$
310 $X_1^s \xleftarrow{\$} \{0,1\}^k$
311 $Auth \leftarrow X_1^s \oplus C_s$
312 $\rho(X_1^{s-1}) \leftarrow Auth$
313 **if** $X_1^s \in \mathrm{Dom}(\rho)$ **then** $bad \leftarrow true$
314 $Auth \xleftarrow{\$} \{0,1\}^k$
315 $\rho(X_1^{t-1}) \leftarrow Auth$
316 **Return** $C \| Auth$

317 **On second query** $C' \| Auth'$
318 $C_0 \| \ldots \| C_{i-1} \| C_i{}' \| \ldots \| C_{t-1}{}' \leftarrow C'$
319 $i \leftarrow \min\{s \mid C_s{}' \neq C_s\}$
320 $Auth \leftarrow \rho(X_1^{i-1}) = X_1^i \oplus C_i$
321 $X_1^{i\prime} \leftarrow Auth \oplus C_i'$
322 **if** $X_1^{i\prime} \in \mathrm{Dom}(\rho)$ **then** $bad \leftarrow true$
323 **if** $i < t-1$ **then**
324 **for** $s \leftarrow i+1$ **to** $t-1$ **do**
325 $X_1^{s\prime} \xleftarrow{\$} \{0,1\}^k$
326 $Auth \leftarrow X_1^{s\prime} \oplus C_s{}'$
327 $\rho(X_1^{s-1\prime}) \leftarrow Auth$
328 **if** $X_1^{s\prime} \in \mathrm{Dom}(\rho)$ **then** $bad \leftarrow true$
329 $Auth \xleftarrow{\$} \{0,1\}^k$
330 $\rho(X_1^{t-1\prime}) \leftarrow Auth$
331 **if** $Auth = Auth'$ **then** $bad \leftarrow true$
332 **Return** 0

Game G4

400 **Given** $\mathtt{M}_0 \| \ldots \| \mathtt{M}_{t-1}$
401 $X_1^{-1} \leftarrow 1^k$
402 **for** $s \leftarrow 0$ **to** $t-1$ **do**
403 $X_0^s \leftarrow s \| 0$
404 **if** $X_0^s \in \mathrm{Dom}(\rho)$ **then** $bad \leftarrow true$
405 $\rho(X_0^s) \leftarrow \mathsf{defined}$
406 $X_1^s \xleftarrow{\$} \{0,1\}^k$
407 $\rho(X_1^{s-1}) \leftarrow \mathsf{defined}$
408 **if** $X_1^s \in \mathrm{Dom}(\rho)$ **then** $bad \leftarrow true$
409 $\rho(X_1^{t-1}) \leftarrow \mathsf{defined}$

410 **Given** $\mathtt{C}' \| \mathtt{Auth}'$
411 $\mathtt{C}_0 \| \ldots \| \mathtt{C}_{i-1} \| \mathtt{C}_i{}' \| \ldots \| \mathtt{C}_{t-1}{}' \leftarrow \mathtt{C}'$
412 $i \leftarrow \min\{s \mid \mathtt{C}_s{}' \neq \mathtt{C}_s\}$
413 $Auth \leftarrow X_1^i \oplus \mathtt{C}_i$
414 $X_1^{i\prime} \leftarrow Auth \oplus \mathtt{C}_i' = X_1^i \oplus \delta$, some $\delta \neq 0$
415 **if** $X_1^{i\prime} \in \mathrm{Dom}(\rho)$ **then** $bad \leftarrow true$
416 **if** $i < t-1$ **then**
417 **for** $s \leftarrow i+1$ **to** $t-1$ **do**
418 $X_1^{s\prime} \xleftarrow{\$} \{0,1\}^k$
419 $\rho(X_1^{s-1\prime}) \leftarrow \mathsf{defined}$
420 **if** $X_1^{s\prime} \in \mathrm{Dom}(\rho)$ **then** $bad \leftarrow true$
421 $Auth \xleftarrow{\$} \{0,1\}^k$
422 $\rho(X_1^{t-1\prime}) \leftarrow \mathsf{defined}$
423 **if** $Auth = \mathtt{Auth}'$ **then** $bad \leftarrow true$

Fig. 8. INT-CTXT-1 Games G3-G4.

A.2 Indistinguishable from Random

In Proposition 1, we measured the indistinguishability between the schemes $\mathcal{E}_{\mathrm{Fun}}$ and $\mathcal{E}_{\mathrm{Rand}}$ under chosen plaintext attacks. The randomized scheme $\mathcal{E}_{\mathrm{Rand}}$ as you recall took any message M that was a multiple of k-bits (k the security parameter) say t and returned a random string of $(t+1)k$-bits. Formally we define $\mathcal{E}_{\mathrm{Rand}}$ as

$$\begin{array}{l}\underline{\text{Algorithm } \mathcal{E}_{\mathrm{Rand}}(M)} \\ \quad M_0 \| \ldots \| M_{t-1} \leftarrow M,\ |M_i| = k \\ \quad \mathrm{Rand} \xleftarrow{\$} \{0,1\}^{(t+1)k} \\ \quad \textbf{Return } \mathrm{Rand}.\end{array}$$

For the definition of indistinguishable from random to make sense in our setting, we give the adversary an additional auxiliary input.

Definition 3. (Indistinguishable from Random): *Let $\mathcal{SE}_{\mathrm{Fun}}$ be the symmetric encryption scheme in Figure 6 using random functions and A_{cpa} a PPT adversary with access to two oracles, $\mathcal{E}_{\mathrm{Fun}}$ and $\mathcal{E}_{\mathrm{Rand}}$. Consider the following experiment with $k \in \mathbb{N}$ and $z \in \{0,1\}^{q(k)}$ for some polynomial q*

$$\begin{array}{l}\text{Experiment } \mathbf{Exp}^{\text{ind\$-cpa}}_{\mathcal{SE}_{\mathrm{Fun}},A_{\mathrm{cpa}}}(k,z)\\ \quad \mathrm{Fun} \xleftarrow{\$} \mathsf{Fun}(k)\\ \quad b \leftarrow A_{\mathrm{cpa}}^{\mathcal{E}_{\mathrm{Fun}},\mathcal{E}_{\mathrm{Rand}}}\\ \quad \text{Return } b\end{array}$$

We denote the winning probability in the adversary breaking IND$-CPA as

$$\mathbf{Adv}^{\text{ind\$-cpa}}_{\mathcal{SE}_{\mathrm{Fun}},A_{\mathrm{cpa}}}(k,z) := \Pr[\mathbf{Exp}^{\text{ind\$-cpa}}_{\mathcal{SE}_{\mathrm{Fun}},A_{\mathrm{cpa}}}(k,z)=1]$$

with the maximum over all possible PPT adversaries as

$$\mathbf{Adv}^{\text{ind\$-cpa}}_{\mathcal{SE}_{\mathrm{Fun}}}(k,q_e,\eta_e,z) := \max_{A_{\mathrm{cpa}}}\{\mathbf{Adv}^{\text{ind\$-cpa}}_{\mathcal{SE}_{\mathrm{Fun}},A_{\mathrm{cpa}}}(k,z)\}$$

where q_e denotes the maximum number of oracle calls to $\mathcal{E}_{\mathrm{Fun}}$ or $\mathcal{E}_{\mathrm{Rand}}$, and η_e the maximum number of k-bit blocks per encryption query.

Proposition 3 *Let $\mathcal{SE}_{\mathrm{Fun}}$ be the authenticated encryption scheme given in Figure 6 using random functions and z any polynomial bounded string in k with $q_e = 1$, and $k \geq 1$. Then*

$$\mathbf{Adv}^{\text{ind\$-cpa}}_{\mathcal{SE}_{\mathrm{Fun}}}(k,q_e,\eta_e,z) \leq \frac{1}{2}(3\eta_e^2+\eta_e)2^{-k}$$

Proof: We can bound the IND$-CPA advantage using game G2 in Figure 7 if we remove the single authentication query. This simulates both $\mathcal{SE}_{\mathrm{Fun}}$ and $\mathcal{SE}_{\mathrm{Rand}}$, which are identical until *bad* is set. Therefore, using the Fundamental Lemma of Game-Playing we have $\mathbf{Adv}^{\text{ind\$-cpa}}_{\mathcal{SE}_{\mathrm{Fun}}}(k,q_e,\eta_e,z) \leq \Pr[\text{Game 2 sets } bad]$. Following the same arguments as used in Proposition 2 (including the assumption that A is deterministic and computationally unbounded), we may transform game G2 to G4. Since for any fixed chain of queries there are at most η_e+1 non-random variables $X_1^{-1}, X_0^{s=0,\ldots,\eta_e-1}$, that do not collide with one another and η_e independent random variables $X_1^{s=0,\ldots,\eta_e-1}$, in $\mathrm{Dom}(\rho)$, it follows that the setting of *bad* in game G4 is bounded above by

$$\Pr[\text{Game G4 sets } bad] \leq \left\{\binom{2\eta_e+1}{2} - \binom{\eta_e+1}{2}\right\}2^{-k}$$

which holds for any computationally unbounded adversary. Therefore, it follows that

$$\begin{aligned}\mathbf{Adv}^{\text{ind\$-cpa}}_{\mathcal{SE}_{\mathrm{Fun}}}(k,q_e,\eta_e,z) &\leq \Pr[\text{Game G4 sets } bad]\\ &\leq \left\{\binom{2\eta_e+1}{2} - \binom{\eta_e+1}{2}\right\}2^{-k}\\ &= \frac{1}{2}(3\eta_e^2+\eta_e)2^{-k}.\end{aligned}$$

□