# Lagrange-DFA Offline Verification Scheme

Alex Jiang [*]

Department of Mathematics, University of California, Berkeley

June 2025

## 1 Introduction

This document describes an adaptable cryptographic scheme for offline verification of whether data associated with pre-encoded identifiers matches pre-encoded secret patterns. The scheme is designed such that if and only if the data associated with an identifier matches the associated pre-encoded regular expression, a pre-defined signal can correctly be decrypted, otherwise any incorrect identifier or data produces a decryption failure indistinguishable from random failure.

## 2 Notation

Let $p_1, p_2$ be large primes. Let $I$ be the set of identifiers $i \in \mathbb{Z}_{p_1} \setminus \{0\}$ whose data $d_i$ should be verified against a regular expression $R_i$. Let $\mathbf{AES\_CTR}_k(m)$ denote AES in CTR mode with key $k$, message $m$, and a null IV. Let $\mathbf{HASH}(m)$ denote a hash function whose digest size is at least as large as $p_2$. Upon a successful verification of $i$ against $R_i$, we would like to be able to decrypt a signal $s_i$, which was encrypted with a key $k_i$, by somehow retrieving $k_i$ and computing $plaintext = \mathbf{AES\_CTR}_{k_i}(s)$.

Let $g \in \mathbb{Z}_{p_1} \setminus \{0\}$ be arbitrary and assume the discrete logarithm problem is hard. Let $L_k : \mathbb{Z}_{p_k} \to \mathbb{Z}_{p_k}$ denote a Lagrange polynomial over $\mathbb{F}_{p_k}$ whose coefficients are also mod $p_k$. Let $M$ be the set of DFA representations of all $R_i$ and be defined by $\{(Q_i, \Sigma, \delta_i, q_{0_i}, F_i)\}$, where $Q_i \subseteq \mathbb{Z}_{p_2}$ is a set of states, $\Sigma = \{0, 1\}^8$ is the byte alphabet, $\delta_i : Q_i \times \Sigma \to Q_i$ is the transition function, $q_{0_i}$ is the initial state, and $F_i \subseteq Q_i$ are accepting states. Let $Q_0$ denote the set of all $q_{0_i}$. Let $\|$ denote bit concatenation.

---

[*]Electronic address: `jiang.alexander@berkeley.edu`

# 3 Construction

For each valid transition in a DFA $M_i$, given by a state $q$, byte $c$, and next state $q'$, compute the point $(\mathbf{HASH}(q \,\|\, c), q') \mod p_2$. Collect these points into the set $T_i$. The set of states $Q_i$ is such that $\forall j, k \in I$, $Q_j \cap Q_k = \emptyset$. The set of accepting states $F_i$ is such that its single element is the encryption key $k_i$. Construct $L_2$ by interpolating over $\bigcup_{i \in I} T_i$. Note that $L_2$ does not contain any information about starting states or accepting states.

For each $i$, compute its commitment $c_i = g^i \mod p_1$. Suppose the encrypted signal $s_i$ is identified by some $d_{s_i}$. Construct $L_1$ by interpolating over $\{(c_i, q_{0_i} \,\| \, d_{s_i}), (d_{s_i}, s_i)\}$. $q_{0_i}$ and $d_{s_i}$ are carefully selected such that $q_{0_i}$ and $d_{s_i}$ fit into the most and least significant halves of $q_{0_i} \,\|\, d_{s_i}$, respectively, with zero-padding as needed.

# 4 Usage

Suppose $j$ is an identifier associated with data $d_j$. Note that $j$ is not necessarily an element of $I$. Compute $d_{intermediary} = L_1(c_j)$ and extract arbitrary values $q_{0_j}$ and $d_{s_j}$ from the most and least significant halves of $d_{intermediary}$, respectively. Compute $s_j = L_1(d_{s_j})$. Regardless of whether $q_{0_j}$ is an element of the true $Q_0$, or whether $s_j$ is a true signal, treat them as such. Initialize $q \leftarrow q_{0_j}$. For each byte $c$ in $d_j$, update $q \leftarrow L_2(\mathbf{HASH}(q \,\|\, c))$. Finally, attempt to decrypt $s_j$ by computing $plaintext = \mathbf{AES\_CTR}_q(s_j)$.