A White-Box Encryption Scheme using Physically Unclonable Functions

Sandra Rasoamiaramanana[1,2], Marine Minier[1] and Gilles Macario-Rat[2]
[1]*Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France*
[2]*Orange Labs, Applied Crypto Group, Châtillon, France*

Keywords: White-Box Cryptography, Code Lifting Attack, Device Binding, Physically Unclonable Functions.

Abstract: When a cryptographic algorithm is executed in a potentially hostile environment, techniques of White-Box Cryptography are used to protect a secret key from a fully-privileged adversary. However, even if the adversary is not able to extract the secret key from the implementation, they might lift the entire white-box code and execute it (this is called a code lifting attack). In this work, we introduce an encryption scheme that can be implemented on an untrusted environment and is still secure even if the white-box code has been lifted. We base our proposal on a Physically Unclonable Function (PUF) to ensure the execution context of our so-called PUF-based encryption scheme. This way, the encryption is "locked" by a particular device.

1 INTRODUCTION

White-Box Cryptography (WBC) was first introduced by Chow et al. in 2002 (Chow et al., 2002a; Chow et al., 2002b) as an obfuscation technique for a cryptographic algorithm. Thus, the purpose of WBC is to produce an implementation of a cryptographic algorithm which hides the key in the data such that a fully-privileged adversary which controls the execution environment of the algorithm and has access to the implementation cannot extract the key from it. This statement refers to the key extraction security. However, Chow et al. also pointed out that in this context, an adversary might lift the implementation and execute the algorithm without knowing the key. Thus, the security of an implementation in the white-box model brings together the resistance to key extraction and the resistance to a code lifting attack. Two notions are generally used to capture those security goals: unbreakability and incompressibility/space-hardness (Delerablée et al., 2013; Biryukov et al., 2014; Fouque et al., 2016; Bogdanov et al., 2016). While unbreakability defines the basic requirement for a white-box implementation which is to merge the secret key inside the implementation, space-hardness ensures that an adversary having a portion of the code cannot execute the algorithm and aims to mitigate code lifting without preventing it. Following this approach, white-box dedicated block ciphers such as SPACE or SPNbox have been proposed in (Biryukov et al., 2014; Bogdanov and Isobe, 2015; Bogdanov et al., 2016; Fouque et al., 2016; Cho et al., 2017). Yet, if the adversary is able to copy the white-box code, they can take advantage of the code functionality without any restriction.

In this work, we address this issue and attempt to avoid code lifting by binding the white-box code to its environment. The idea is that even if the adversary lifts the entire white-box code, they will not take advantage of the code functionality. For that, we make use of a *Physically Unclonable Function* (PUF) (Maes, 2012) to reinforce the security against code lifting. A PUF is a physical system that can be challenged with a stimuli and which reacts: the so-called *response* depends on manufacturing variations which induce a structural disorder that cannot be cloned or reproduced exactly. Thanks to such a device-specific behavior, a PUF might be used for identification, authentication, and even key generation.

We formalize the concept of PUF in Section 2. In Section 3, we introduce the notion of *PUF-based encryption scheme* as an encryption scheme which takes three inputs: a secret key, a plaintext and a PUF response. Thus, the PUF response adds a variability mechanism to the encryption scheme. First, we formalize the notion of *lockability*[1] to guarantee the security of the scheme when implemented in the white-box model: lockability states the difficulty of executing a lifted white-box code on a different device. Then, we instantiate the scheme with the *tweakable Even-Mansour* construction of (Cogliati et al., 2015) and

[1]Neologism to define the possibility to lock a cipher to an execution environment.

Rasoamiaramanana, S., Minier, M. and Macario-Rat, G.
A White-Box Encryption Scheme using Physically Unclonable Functions.
DOI: 10.5220/0009781002790286
In *Proceedings of the 17th International Joint Conference on e-Business and Telecommunications (ICETE 2020) - SECRYPT*, pages 279-286
ISBN: 978-989-758-446-6

the white-box dedicated block cipher SPNbox-8 (Bogdanov et al., 2016).

2 PHYSICALLY UNCLONABLE FUNCTIONS

This section gives an overview of the concept of PUF without being exhaustive. We refer to (Maes, 2012) for a complete description and analysis.

2.1 Characterization of a Physically Unclonable Function

A PUF is a physical entity (e.g. an integrated circuit) that presents random imperfections in its structure. The imperfections result from the manufacturing process and are expected to be instance specific and unclonable. The term PUF also refers to the "expression" (the challenge-response behavior) of these random imperfections. To characterize a PUF, the physical disorder (e.g. a current or a voltage) is generally measured under a specific condition (stimuli) and converted into digital form. Thus, a PUF performs a probabilistic functional operation: given a certain input (a stimuli, also called a *challenge*) it produces a measurable output (the *response*). Consequently, a PUF is seen as a physical realization of a probabilistic mapping $\mathsf{puf}\colon \mathcal{X} \to \mathcal{Y}$ where $\mathcal{X}$ is the domain space called the challenge set and $\mathcal{Y}$ is an output range called the response set. The creation of a PUF is expressed by invoking a manufacturing process on a set of parameters param. We write $\mathsf{puf} \leftarrow_{\$} \textsc{Create}(\mathsf{param})$ to signify that a PUF instance is created. The set of all instances created according to a set of parameters is called a PUF class and denoted by $\mathcal{P}$. An instance of $\mathcal{P}$ can be evaluated on a challenge $x \in \mathcal{X}$ using an evaluation procedure EVAL: we write $y \leftarrow \textsc{Eval}(\mathsf{puf}, x)$ or simply $y \leftarrow \mathsf{puf}(x)$ for the evaluation of puf on the challenge x. The outputs of a PUF instance are generally noisy, i.e., puf outputs two distinct responses y, y' when evaluated twice on the same challenge x. Thus, a PUF response is considered as a random variable, and information about the distribution of the PUF responses is obtained through experiment. An experiment on a PUF class is parametrized by t, q_c, and q_r which are the number of evaluations of one instance on the same challenge, the number of created instances, and the number of distinct evaluations on an instance, respectively. The experiment gives estimated descriptive statistics of the *intra-distance* δ_1, of the *instance inter-distance* δ_2 and of the PUF *inter-distance* δ_3. The *intra-distance* is the distance between two distinct responses of an instance evaluated on the same challenge. In the same manner, the *instance inter-distance* and the PUF *inter-distance* characterize the distance between any two responses y, y' of an instance puf evaluated on two distinct challenges x, x' and the distance between two outputs y, y' of two distinct instances puf, puf' evaluated on the same challenge x, respectively. On the one hand, the intra-distance property estimates the *reproducibility* of a PUF output. Thus, low intra-distance ensures that the PUF outputs are highly reproducible. On the other hand, the PUF inter-distance estimates the *uniqueness* of an instance's output while the instance inter-distance ensures the uniqueness of a response given a challenge. The intra-distance δ_1 should be lower than δ_2 and δ_3 to uniquely distinguish outputs from different instances and outputs from different challenges. For the rest of the paper, we denote by dist the distance. Another property of PUF outputs that should be estimated is the *min-entropy*. The min-entropy of a binary string represents the number of uniform bits and measures the uncertainty one has on the output. The *conditional min-entropy* is more relevant for the distribution of PUF outputs since an adversary should not deduce enough information about an unknown output given many outputs. These properties characterize the output distribution of PUF instances and give an idea of the reliability of a PUF class. A PUF class may verify other properties (Maes, 2012). We focus on the *unclonability* and *unforgeability* properties.

2.2 Security Notions

We use the formal definitions of unforgeability and unclonability that have been introduced in the paper of Armknecht et al. (Armknecht et al., 2016).

We denote by $\mathcal{A}$ a Probabilistic Polynomial Time (PPT) adversary. If λ is the security parameter then the running time of $\mathcal{A}$ is polynomial in λ. Let $\varepsilon\colon \mathbb{N} \to \mathbb{R}$ be a negligible function, i.e. for any polynomial function $p\colon \mathbb{N} \to \mathbb{N}$ there exists n_0 such that $\varepsilon(n) < 1/p(n)$ for any $n > n_0$. The following unforgeability game states the advantage of an adversary $\mathcal{A}$ to guess one output of a PUF without having access to the device. We assume that the challenger has access to the manufacturing process and thus can invoke the creation procedure on a set of parameters param. The adversary can only choose a set of parameters to obtain new instances from an oracle. Thus, they can get instances of one PUF class or instances of different PUF classes depending on the chosen set of parameters. We assume that there exists a recovery procedure that recovers a noise-free response from noisy ones. Thus, the adversary is only required to guess a noisy response at a

distance not more than δ_1 from the correct response.

Set Up. The challenger selects a manufacturing process and initial parameters param. The challenger sends $(1^\lambda, \mathsf{param})$ to the adversary $\mathcal{A}$ and initializes two counters c_0, c_1.

Learning. $\mathcal{A}$ adaptively issues two types of oracle query: a creation query to create a new instance under some chosen parameters and a response query to get the output of an instance of their choice. $\mathcal{A}$ is allowed to issue at most q_c creation queries and q_r response queries.

- When $\mathcal{A}$ issues a creation query with param', the challenger creates $\mathsf{puf}_{c_0} \leftarrow \textsc{Create}(\mathsf{param})$ if $\mathsf{param}' = \mathsf{param}$ and increments c_0, or creates $\mathsf{puf}'_{c_1} \leftarrow \textsc{Create}(\mathsf{param}')$ if $\mathsf{param}' \neq \mathsf{param}$ but is a valid creation parameter and increments c_1. Otherwise the challenger responds $\perp$.
- When $\mathcal{A}$ issues a response query with (b, i, x_j), the challenger sends $y_{i,j} \leftarrow \mathsf{puf}_i(x_j)$ if $b = 0$ and $i \leqslant c_0$ or sends $y'_{i,j} \leftarrow \mathsf{puf}'_i(x_j)$ if $b = 1$ and $i \leqslant c_1$. Otherwise the challenger returns $\perp$.

Guess. $\mathcal{A}$ outputs (i^*, x^*, y^*).

The index i^* denotes the instance chosen by $\mathcal{A}$ such that y^* is probably the result of the evaluation of puf_{i^*} on x^*. We denote by EUF-CICA the *Existential Unforgeability under Chosen Instance and Challenge Attack* defined as follows.

Definition 1 (EUF-CICA)**.** *Let* $\mathsf{Adv}^{\textit{euf-cica}}_{\mathcal{A},\mathcal{P}}(\lambda, \delta_1)$ *be the advantage of an adversary* $\mathcal{A}$ *playing the unforgeability game. We have* $\mathsf{Adv}^{\textit{euf-cica}}_{\mathcal{A},\mathcal{P}}(\lambda, \delta_1) =$

$$\Pr\left[\mathit{dist}(y^*, \mathit{puf}_{i^*}(x^*)) \leqslant \delta_1 \,|\, x^* \in \mathcal{X} \setminus \mathcal{X}_{i^*}\right] - \frac{|B|}{|\mathcal{Y}|}$$

where puf_{i^*} *is a PUF instance that has been created by the challenger in the learning phase,* $\mathcal{X}_{i^*}$ *is the set of challenges that have been issued for the instance* i^* *and* $B = \{y \mid y_{i^*} \leftarrow \mathit{puf}_{i^*}(x^*)$ *and* $\mathit{dist}(y_{i^*}, y) \leqslant \delta_1\}$ *is the set of outputs* y *that are at a distance at most* δ_1 *from* $\mathsf{puf}_{i^*}(x^*)$.

A PUF provides $(q_c, q_r, \delta_1, \varepsilon)$*-EUF-CICA security if for any PPT adversary* $\mathcal{A}$ *we have*

$$\Pr\left[\mathsf{Adv}^{\text{euf}-\text{cica}}_{\mathcal{A},\mathcal{P}}(\lambda, \delta_1) > 0\right] \leqslant \varepsilon(\lambda).$$

The adversary wins the game if their guess y^* belongs to the set B with probability higher than randomly picking an element of B.

A stronger notion is unclonability which expresses the hardness of constructing two PUF instances that show the same input-output behavior. The unclonability game is as follows.

Set Up. The setup phase is the same as in the unforgeability game.

Learning. $\mathcal{A}$ issues oracle queries as in the learning phase of the unforgeability game.

Guess. $\mathcal{A}$ outputs (i^*, b, j^*).

The output (i^*, b, j^*) is interpreted as follows: i^* gives the index of an instance created under the paramater param. If $b = 0$ then j^* is the index of an instance created under the parameter param; otherwise j^* is the index of an instance created under the parameter param'. To win, $\mathcal{A}$ should exhibit two clones either from the same class or from two distinct classes. In other words, the unclonability should garantee that it is both hard to construct two clones using the same set of parameters and to find a set of parameters that enables the construction of clones.

We denote by UUC-CICA the *Universal Unclonability* security defined by the following.

Definition 2 (UUC-CICA)**.** *Let* $\mathsf{Adv}^{\text{uuc}-\text{cica}}_{\mathcal{A},\mathcal{P}}(\lambda, \delta_1)$ *be the advantage of an adversary* $\mathcal{A}$ *playing the unclonability game. Let* puf_{i^*} *and* puf'_{j^*} *be the PUF instances which correspond to the output* (i^*, b, j^*) *of* $\mathcal{A}$*. We have* $\mathsf{Adv}^{\text{uuc}-\text{cica}}_{\mathcal{A},\mathcal{P}}(\lambda, \delta_1) =$

$$\Pr\left[\forall x \in \mathcal{X}, \mathit{dist}(\mathit{puf}_{i^*}(x), \mathit{puf}'_{j^*}(x)) \leqslant \delta_1\right].$$

A PUF provides $(q_c, q_r, \delta_1, \varepsilon)$*-UUC-CICA if for any PPT adversary* $\mathcal{A}$ *we have* $\mathsf{Adv}^{\text{uuc}-\text{cica}}_{\mathcal{A},\mathcal{P}}(\lambda, \delta_1) \leqslant \varepsilon(\lambda)$.

3 GENERAL CONSTRUCTION

In the following, we introduce the PUF-based encryption scheme which is used to define a PUF-based white-box encryption scheme when provided with a white-box compiler.

3.1 PUF-based Encryption Scheme

In the definition of a *PUF-based encryption scheme* below, the encryption and the decryption algorithms take three arguments, the additional argument being the PUF output. We denote by $\mathcal{M}$ and $\mathcal{C}$ the sets of plaintexts and ciphertexts, respectively.

Definition 3 (PUF-based Encryption Scheme (PES))**.** *Let* $\mathcal{P} = \{\mathit{puf}\colon \mathcal{X} \to \mathcal{Y}\}$ *be a PUF class. A PUF-based encryption scheme consists of polynomial time algorithms* $(\mathsf{KGen}, \textsc{eval}, \mathsf{Enc}, \mathsf{Dec})$ *such that*

- KGen *is a probabilistic algorithm which outputs a key:* $K \leftarrow_\$ \mathsf{KGen}(1^\lambda)$.
- Eval *is a probabilistic algorithm that evaluates a PUF instance:* $y \leftarrow_\$ \textsc{eval}(\mathit{puf}, x)$.
- Enc *is a probabilistic algorithm:* $C \leftarrow_\$ \mathsf{Enc}(K, y, M)$.

- Dec *is a deterministic algorithm:* $M \leftarrow \mathsf{Dec}(K,y,C)$.

A PUF-based encryption scheme satisfies correctness for an instance puf *if* $\forall M \in \mathcal{M}$ *and* $\forall x \in \mathcal{X}$, $\Pr[\mathsf{Dec}(K,y,\mathsf{Enc}(K,y,M)) = M \mid y \leftarrow \mathsf{puf}(x)] = 1$.

Definition 4 (White-Box Compiler)**.** *Let* $\mathcal{E} = (\mathsf{KGen},\mathsf{Enc},\mathsf{Dec})$ *be a symmetric encryption scheme. A white-box compiler for* $\mathcal{E}$ *is a probabilistic algorithm* Comp *that outputs a white-box program* P *of the cipher for a fixed key* K.

- *If* P *implements* Enc_K *then* $\forall M \in \mathcal{M}$, $\Pr[\mathsf{P}(M) = \mathsf{Enc}(K,M)] = 1$. *We say that* $(\mathsf{KGen},\mathsf{Enc},\mathsf{Dec},\mathsf{Comp})$ *is a white-box encryption scheme.*
- *If* P *implements* Dec_K *then* $\forall C \in \mathcal{C}$, $\Pr[\mathsf{P}(C) = \mathsf{Dec}(K,C)] = 1$. *We say that* $(\mathsf{KGen},\mathsf{Enc},\mathsf{Dec},\mathsf{Comp})$ *is a white-box decryption scheme.*

If Comp is a white-box compiler defined for the PES of Definition 3, then the white-box program P is such that, for any y and M, $\mathsf{P}(y,M) = \mathsf{Enc}(K,y,M)$ in the case of a white-box encryption scheme and $\mathsf{P}(y,C) = \mathsf{Dec}(K,y,C)$ for a white-box decryption scheme. The PES and the white-box compiler form a PUF-based white-box encryption scheme (PWE).

3.2 Security Models

We consider the white-box setting and state the security of a PWE. We consider a white-box implementation of the encryption algorithm, i.e. the program P outputted by the white-box compiler implements Enc_K for a fixed key K. The adversary $\mathcal{A}$ is divided into two adversaries $(\mathcal{A}_1,\mathcal{A}_2)$ such that $\mathcal{A}_2$ is run on $\mathcal{A}_1$'s output. $\mathcal{A}_1$ is allowed to make at most q_c creation queries and q_r response queries while $\mathcal{A}_2$ is allowed to make at most q decryption queries.

3.2.1 Unbreakability

In the unbreakability game, we only consider the adversary $\mathcal{A}_2$ whose goal is to recover the secret key K. We consider the Chosen-Plaintext Attack (CPA) where the adversary chooses q plaintexts (and the input y) and encrypts them with the program P and the Chosen-Ciphertext Attack (CCA) where they are allowed to query a decryption oracle with q chosen ciphertexts (and input y) in addition to q chosen-plaintexts. The advantage of the adversary $\mathcal{A}_2$ in the $\mathsf{UBK\text{-}ATK}$ security game is the probability that $\mathcal{A}_2$ outputs the key K. As in (Delerablée et al., 2013), the white-box encryption scheme is secure in the sense of $\mathsf{UBK\text{-}ATK}$ for $\mathsf{ATK} \in \{\mathsf{CPA},\mathsf{CCA}\}$, if the advantage of any PPT adversary is negligible.

Moreover, even if full break is not possible, the adversary can lift the code and encrypt without knowing the key. We define the notion of lockability to capture the difficulty of executing a lifted white-box code without a PUF instance.

3.2.2 Lockability

We assume that the adversary has access to the white-box program P. Their goal is to correctly encrypt a random plaintext given a random challenge x. $\mathcal{A}_1$ plays an $\mathsf{EUF\text{-}CICA}$ or an $\mathsf{UUC\text{-}CICA}$ game while the attack model for the adversary $\mathcal{A}_2$ is either CPA or CCA.

We define two security notions, depending on the attack model used against the PUF: either a forging attack or a cloning attack.

We write $\mathsf{LCK\text{-}FORGE}$ for the lockability security under CPA or CCA which is described by the following security game.

Set Up. The challenger selects a manufacturing process and initial parameters param. They generate $K \leftarrow_{\$} \mathsf{KGen}(1^\lambda)$ uniformly at random and compile $\mathsf{P} \leftarrow_{\$} \mathsf{Comp}(K)$. They send $(1^\lambda,\mathsf{param},\mathsf{P})$ to the adversary and initializes two counters c_0,c_1.

Learning 1. $\mathcal{A}_1$ issues creation queries and response queries. As in the unforgeability game, the challenger creates $\mathsf{puf}_{c_0} \leftarrow \textsc{Create}(\mathsf{param})$ or $\mathsf{puf'}_{c_1} \leftarrow \textsc{Create}(\mathsf{param'})$ when receiving creation queries and sends $y_{i,j} \leftarrow \mathsf{puf}_i(x_j)$ or $y_{i,j} \leftarrow \mathsf{puf}'_i(x_j)$ when receiving response queries.
$\mathcal{A}_1$ outputs (i^*,x^*,y^*) and sends (i^*,x^*) to the challenger. The challenger evaluates $y \leftarrow \mathsf{puf}_{i^*}(x^*)$.

Learning 2. $\mathcal{A}_2$ chooses q plaintexts and encrypts them with P and y^*. In the CCA model, $\mathcal{A}_2$ issues decryption queries with q chosen ciphertexts C_j and gets $M_j \leftarrow \mathsf{Dec}(K,y,C_j)$.

Challenge. The challenger draws $M \leftarrow_{\$} \mathcal{M} \setminus \{M_j\}$ uniformly at random, computes $C \leftarrow \mathsf{Enc}(K,y,M)$ and sends M.

Guess. $\mathcal{A}_2$ outputs C^*.

Definition 5 ($\mathsf{LCK\text{-}FORGE}$)**.** *Let* $\mathcal{P}$ *be a PUF class with an intra-distance* δ_1. *Let* $\mathcal{E}$ *be the PUF-based white-box encryption scheme. Let* $\mathsf{Adv}^{\mathrm{lck-forge}}_{\mathcal{A},\mathcal{E}}(\lambda,\delta_1)$ *be the advantage of an adversary playing the game above. We have* $\mathsf{Adv}^{\mathrm{lck-forge}}_{\mathcal{A},\mathcal{E}}(\lambda,\delta_1) = \Pr\left[C^* = C : \begin{array}{c} C^* \leftarrow \mathcal{A}_2^{\mathcal{O}}(1^\lambda,r,\mathsf{P}) \\ r=(i^*,x^*,y^*)\leftarrow\mathcal{A}_1(1^\lambda,\mathsf{param}) \end{array}\right]$ *where* $\mathcal{O}$ *denotes the decryption oracle in the* CCA *model. A PUF-based white-box encryption scheme* $\mathcal{E}$ *provides*

$(q_c, q_r, q, \varepsilon')$-*LCK-FORGE security if for any PPT adversary* $\mathcal{A}$ *it holds that:*

$$\mathsf{Adv}^{\mathrm{lck-forge}}_{\mathcal{A},\mathcal{E}}(\lambda, \delta_1) \leqslant \varepsilon'(\lambda).$$

In the above security game, the adversary is required to output a PUF challenge x^* for which they can forge a valid output y^* for an instance puf_{i^*}. The second learning phase allows the adversary to interact with a decryption oracle (in the CCA model). If the attack model for $\mathcal{A}_2$ is the CPA model, then in learning 2, $\mathcal{A}_2$ is only allowed to choose at most q plaintexts and to encrypt them with P and y^*. For both models, the adversary wins if $\mathcal{A}_1$ is able to forge a valid pair (x^*, y^*) for a chosen instance i^* or if the pair returned by $\mathcal{A}_1$ is not valid but $\mathcal{A}_2$ guesses C.

Remark 1. *In the CCA model,* $\mathcal{A}_2$ *can compute* $C_j \leftarrow \mathsf{P}(y^*, M_j)$ *and send* C_j *to the decryption oracle. Thus,* $\mathcal{A}_2$ *is able to verify if the guess of* $\mathcal{A}_1$ *is correct. Otherwise,* $\mathcal{A}_1$ *is allowed to restart the learning* 1 *and makes a new guess. This way the attack model is adaptive. However, the total number of queries made by* $\mathcal{A}_1$ *is still bounded by* q_r *and* q_c *and the total number of queries made by* $\mathcal{A}_2$ *is bounded by* q*. Consequently, the advantage of the adversary regarding LCK-FORGE is the same if the model is adaptive or not.*

Theorem 1. *If the PUF class* $\mathcal{P}$ *provides* $(q_c, q_r, \delta_1, \varepsilon)$-*EUF-CICA security then* $\mathcal{E}$ *provides* $(q_c, q_r, q, \varepsilon')$-*LCK-FORGE security with*

- $\varepsilon' \colon \lambda \mapsto \left(1 - \frac{1}{|\mathcal{C}|}\right) \cdot \varepsilon(\lambda) + \frac{1}{|\mathcal{C}|}$ *for CPA.*
- $\varepsilon' \colon \lambda \mapsto \left(1 - \frac{1}{|\mathcal{C}|-q}\right) \cdot \varepsilon(\lambda) + \frac{1}{|\mathcal{C}|-q}$ *for CCA.*

Proof. Let E be the event "$\mathsf{Adv}^{\mathrm{euf-cica}}_{\mathcal{A}_1,\mathcal{P}}(\lambda, \delta_1) > 0$". $\mathsf{Adv}^{\mathrm{lck-forge}}_{\mathcal{A},\mathcal{E}}(\lambda, \delta_1) =$

$$\Pr[C^* = C \mid E] \times \Pr[E] + \Pr\left[C^* = C \,\middle|\, \overline{E}\right] \times \Pr\left[\overline{E}\right]$$

Since $\mathcal{P}$ provides $(q_c, q_r, \delta_1, \varepsilon)$-EUF-CICA security then $\Pr[E] \leqslant \varepsilon(\lambda)$ and $\Pr[C^* = C \mid E] = 1$. This reflects the fact that once the PUF response is guessed correctly, the adversary inevitably finds the correct ciphertext. Besides, $\Pr\left[C^* = C \,\middle|\, \overline{E}\right] = \frac{1}{|\mathcal{C}|}$ in the CPA model and $\Pr\left[C^* = C \,\middle|\, \overline{E}\right] = \frac{1}{|\mathcal{C}|-q}$ in the CCA model.

$$\mathsf{Adv}^{\mathrm{lck-forge}}_{\mathcal{A},\mathcal{E}}(\lambda, \delta_1) \leqslant \left(1 - \frac{1}{|\mathcal{C}|}\right) \cdot \varepsilon(\lambda) + \frac{1}{|\mathcal{C}|}$$

under CPA and,

$$\mathsf{Adv}^{\mathrm{lck-forge}}_{\mathcal{A},\mathcal{E}}(\lambda, \delta_1) \leqslant \left(1 - \frac{1}{|\mathcal{C}|-q}\right) \cdot \varepsilon(\lambda) + \frac{1}{|\mathcal{C}|-q}$$

under CCA. □

The notion of lockability under cloning attack LCK-CLONE expresses the situation in which we verify the lockability except if the adversary exhibits two distinct instances that have the same input-output behavior. The security game for LCK-CLONE under CPA or CCA is described as follows:

Set Up. The set up is the same as in LCK-FORGE.

Learning 1**.** The learning phase is as in LCK-FORGE. $\mathcal{A}_1$ outputs (i^*, b, j^*).

Learning 2**.** $\mathcal{A}_2$ chooses q plaintexts and challenges and encrypts them with P. In the CCA model, in addition $\mathcal{A}_2$ issues decryption queries with q chosen ciphertexts C_j and challenges x_j and gets $M_j \leftarrow \mathsf{Dec}(K, y_j, C_j)$ where $y_j \leftarrow \mathsf{puf}_{i^*}(x_j)$.

Challenge. The challenger picks $M \leftarrow_{\$} \mathcal{M} \setminus \{M_j\}$ and $x \leftarrow_{\$} \mathcal{X} \setminus \{x_j\}$ uniformly at random, computes $C \leftarrow \mathsf{Enc}(K, y, M)$ where $y \leftarrow \mathsf{puf}_{i^*}(x)$ and sends (M, x).

Guess. $\mathcal{A}_2$ outputs C^*.

Definition 6. *Let* $\mathcal{P}$ *be a PUF class with an intra-distance* δ_1*. Let* $\mathcal{E}$ *be the PUF-based white-box encryption scheme. Let* $\mathsf{Adv}^{\mathrm{lck-clone}}_{\mathcal{A},\mathcal{E}}(\lambda, \delta_1)$ *be the advantage of an adversary playing the game above. We have* $\mathsf{Adv}^{\mathrm{lck-clone}}_{\mathcal{A},\mathcal{E}}(\lambda, \delta_1) =$ $\Pr\left[C^* = C \colon \underset{r=(i^*,b,j^*) \leftarrow \mathcal{A}_1(1^\lambda, \mathit{param})}{C^* \leftarrow \mathcal{A}_2^{O}(1^\lambda, r, \mathsf{P})}\right]$ *where* O *denotes the decryption oracle in the CCA model. A PUF-based white-box encryption scheme provides* $(q_c, q_r, q, \delta_1, \varepsilon')$-*LCK-CLONE security if for any PPT adversary* $\mathcal{A}$ *it holds that:* $\mathsf{Adv}^{\mathrm{lck-clone}}_{\mathcal{A},\mathcal{E}}(\lambda, \delta_1) \leqslant \varepsilon'(\lambda)$.

Theorem 2. *If the PUF class* $\mathcal{P}$ *provides* $(q_r, q_c, \delta_1, \varepsilon)$-*UUC-CICA security then the PUF-based white-box encryption scheme provides* $(q_c, q_r, q, \delta_1, \varepsilon')$-*LCK-CLONE security with*

- $\varepsilon' \colon \lambda \mapsto \left(1 - \frac{1}{|\mathcal{C}|}\right) \cdot \varepsilon(\lambda)^{1/(|\mathcal{X}|-q)} + \frac{1}{|\mathcal{C}|}$ *for CPA.*
- $\varepsilon' \colon \lambda \mapsto \left(1 - \frac{1}{|\mathcal{C}|-q}\right) \cdot \varepsilon(\lambda)^{1/(|\mathcal{X}|-q)} + \frac{1}{|\mathcal{C}|-q}$ *for CCA.*

Proof. Let E be the event "$\mathsf{dist}(\mathsf{puf}_{i^*}(x), \mathsf{puf'}_{j^*}(x)) \leqslant \delta_1$". $\mathsf{Adv}^{\mathrm{lck-clone}}_{\mathcal{A},\mathcal{E}}(\lambda, \delta_1) =$

$$\Pr[C^* = C \mid E] \times \Pr[E] + \Pr\left[C^* = C \,\middle|\, \overline{E}\right] \times \Pr\left[\overline{E}\right]$$

Since, $\mathcal{P}$ provides $(q_r, q_c, \delta_1, \varepsilon)$-UUC-CICA security then $\Pr[E] \leqslant \varepsilon(\lambda)^{1/(|\mathcal{X}|-q)}$ and $\Pr[C^* = C \mid E] = 1$. $\Pr\left[C^* = C \,\middle|\, \overline{E}\right] = \frac{1}{|\mathcal{C}|}$ in the CPA model and $\Pr\left[C^* = C \,\middle|\, \overline{E}\right] = \frac{1}{|\mathcal{C}|-q}$ in the CCA model. Hence,

$$\mathsf{Adv}^{\mathrm{lck-clone}}_{\mathcal{A},\mathcal{E}}(\lambda, \delta_1) \leqslant \left(1 - \frac{1}{|\mathcal{C}|}\right) \cdot \varepsilon(\lambda)^{\frac{1}{(|\mathcal{X}|-q)}} + \frac{1}{|\mathcal{C}|}$$

under CPA and,

$$\mathsf{Adv}^{\mathrm{lck-clone}}_{\mathcal{A},\mathcal{E}}(\lambda,\delta_1) \leqslant \left(1-\frac{1}{|C|-q}\right)\cdot\varepsilon(\lambda)^{\frac{1}{(|X|-q)}} + \frac{1}{|C|-q}, \text{ under CCA.}$$

□

3.3 Construction of a PUF-based Block Cipher

In this section, we provide one instantiation of the encryption scheme using a tweakable block cipher. A tweakable block cipher takes three inputs: the plaintext, the secret key, and a public value called the tweak. We refer to (Liskov et al., 2002) for more details. Thus, it matches the Definition 3 when combined with a PUF class. We use the 2 rounds Tweakable Even-Mansour (2-TEM) of (Cogliati et al., 2015). This construction applies an almost XOR universal (AXU) hash function to the tweak value. Since the tweak inputs are noise-free PUF responses, they are obtained after a recovery procedure. Thereby, the hash function takes a noise-free but non-uniform input y and transforms it to a nearly uniform value xored with the plaintext.

PES with 2 Rounds of TEM. Let $\mathcal{P}=\{\mathsf{puf}\colon \mathcal{X}\to\mathcal{Y}\}$ be a PUF class. Let $\mathcal{H}=\{H_K\colon \mathcal{Y}\to\mathbb{F}_2^n\}_{K\in\mathcal{K}}$ be a family of ρ_1-uniform and ρ_2-AXU hash functions indexed by a set of keys $\mathcal{K}$. Let $\mathsf{Enc}\colon \mathcal{K}\times\mathcal{Y}\times\mathbb{F}_2^n\to\mathbb{F}_2^n$ be a 2-TEM block cipher (see Figure 1). For any key $K=(K_1,K_2)$, H_{K_1},H_{K_2} are two hash functions of the family $\mathcal{H}$ and P_1,P_2 are two public permutations. For any PUF response $y\in\mathcal{Y}$ corresponding to a challenge $x\in\mathcal{X}$, the encryption function is defined, for any plaintext $M\in\mathbb{F}_2^n$, by

$$\mathsf{Enc}(K,y,M) = P_2(P_1(M\oplus H_{K_1}(y))\oplus H_{K_1}(y) \oplus H_{K_2}(y))\oplus H_{K_2}(y).$$

In the same way, the decryption function is defined for any ciphertext $C\in\mathbb{F}_2^n$ by

$$\mathsf{Dec}(K,y,C) = P_1^{-1}(P_2^{-1}(C\oplus H_{K_2}(y)) \oplus H_{K_2}(y)\oplus H_{K_1}(y))\oplus H_{K_1}(y).$$

We denote by 2-TEM$(\mathcal{P})=(\mathbb{F}_2^{2n},\mathbb{F}_2^n,\mathbb{F}_2^n,\mathsf{Enc},\mathsf{Dec},\mathcal{P})$ the constructed PES.

Keyed Hash Functions. We construct the following family of keyed hash functions:

Definition 7. *Let $E\colon \mathbb{F}_2^n\times\mathbb{F}_2^n\to\mathbb{F}_2^n$ be a block cipher. Let $\mathcal{H}=\{H_K\colon (\mathbb{F}_2^n)^2\to\mathbb{F}_2^n\}_{K\in\mathbb{F}_2^n}$ be a family of hash functions indexed by a set of keys where any hash function H_K is defined as follows: for any $x=(x_1,x_2)$, $H_K(x)=E_K(x_1)\oplus E_{K^2}(x_2)$ with $K^2=K\otimes K$ is a multiplication in the finite field $\mathbb{F}_{2^n}$.*

Remark 2. *The function $K\mapsto K^2$ is a permutation in $\mathbb{F}_{2^n}$, thus E_{K^2} is a permutation.*

We show that the above family of hash functions is almost uniform and XOR universal.

Definition 8. *Let $E\colon \mathbb{F}_2^n\times\colon \mathbb{F}_2^n\to\mathbb{F}_2^n$ be an r rounds iterative block cipher and let $F^i_{K_i}$ be the i-th round function with a round key K_i derived from a master key K.*

- *A differential characteristic is a sequence of $r+1$ differences $(\Delta_0,\ldots,\Delta_r)$ where Δ_0 is the difference between two inputs and Δ_r is the output difference.*
- *For a key K, the fixed-key differential characteristic probability π_K is the probability for a pair of inputs to follow a given differential characteristic:* $\pi_K=\Pr_X\left[F^i_{K_i}(X)\oplus F^i_{K_i}(X\oplus\Delta_0)=\Delta_i, 1\leqslant i\leqslant r\right]$.
- *The expected differential characteristic probability π is the differential characteristic probability averaged over all round keys:* $\pi=\Pr_{X,K}\left[F^i_{K_i}(X)\oplus F^i_{K_i}(X\oplus\Delta_0)=\Delta_i, 1\leqslant i\leqslant r\right] = \mathbb{E}_K[\pi_K]$.

Let Π denotes an upper bound on the expected differential characteristic probability of the block cipher E introduced in Definition 7.

Theorem 3. *The family of Definition 7 is $\frac{34}{2^n}$-uniform and Π-AXU.*

Proof. We first prove the XOR universality. Let $x\neq x'\in(\mathbb{F}_2^n)^2$ and $b\in\mathbb{F}_2^n$. Without loss of generality, we assume that $x_1\neq x'_1$. Otherwise, interchange the subscript 1 with the subscript 2 in the following. Let $A=\{K\in\mathbb{F}_2^n\colon H_K(x)\oplus H_K(x')=b\}$, we have:

$$\Pr_K\left[H_K(x)\oplus H_K(x')=b\right]=\frac{|A|}{2^n} \text{ with } |A| = \left|\{K,K^2\colon E_K(x_1)\oplus E_{K^2}(x_2)\oplus E_K(x'_1)\oplus E_{K^2}(x'_2)=b\}\right| = |B|, \text{ with } |B| = \sum_{K^2}\left|\{K\colon E_K(x_1)\oplus E_K(x'_1)=b\oplus E_{K^2}(x_2)\oplus E_{K^2}(x'_2)\}\right|$$

Since $|\{K\in\mathbb{F}_2^n\colon E_K(x_1)\oplus E_K(x_1\oplus a)=b'|\leqslant\Pi\cdot 2^n$ for any $b'\in\mathbb{F}_2^n$ and $K\mapsto K^2$ is a bijection then $|A|\leqslant\Pi\cdot 2^n$.

Now, we prove that the family $\mathcal{H}$ is $\frac{34}{2^n}$-uniform. Let $x\in(\mathbb{F}_2^n)^2$ and $y\in\mathbb{F}_2^n$. We compute the cardinality of $A=\{K\in\mathcal{K}\colon H_K(x)=E_K(x_1)\oplus E_{K^2}(x_2)=y\}$ given $x=(x_1,x_2)$ and y. As the probability of picking randomly a key is equal to $1/2^n$, it means that the number of keys verifying properties of A follows a Poisson distribution with parameter $\lambda=1$ as it could

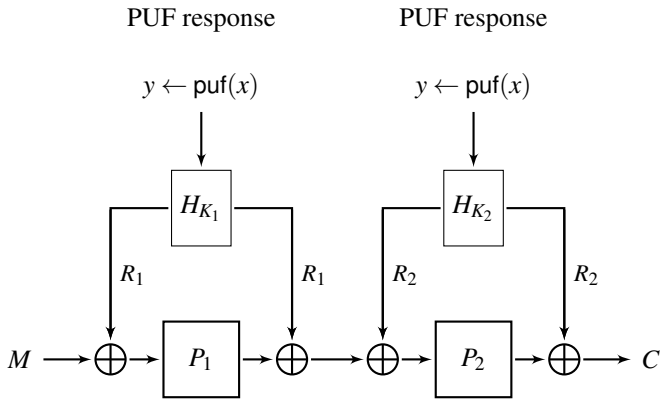

Figure 1: $\mathsf{TEM}(\mathcal{P})$ construction with 2 rounds and PUF responses as tweak inputs.

be considered as a sum of Bernoulli events with a very low probability. Thus, it corresponds with the modelling of the law of rare events. For $n = 128$, we obtain an upper bound equal to 34 for the cardinal of A directly applying the density of the Poisson law. Thus, $\Pr_K[H_K(x) = y] \leqslant \frac{34}{2^n}$. □

Remark 3 (Estimation of the Probability when K is Fixed.)**.** *In the white-box setting, the key K is fixed. Consequently, we need to estimate the XOR universality for a fixed key K. Π is an upper bound on the expected differential characteristic probability of the block cipher E over all keys. When a key K is fixed, we consider the fixed-key differential characteristic probability π_K which is the probability that an input pair $(x, x \oplus a)$ fulfills a differential characteristic $(\Delta_0 = a, \Delta_1, \ldots, \Delta_r = b)$.*

Let $N(a,b) = |\{x\colon E_K(x) \oplus E_K(x \oplus a) = b\}|$, i.e. $N(a,b)$ is the number of pairs (x,x') that fulfill the differential characteristic $(\Delta_0 = a, \Delta_1, \ldots, \Delta_r = b)$ for a fixed key K. Let q_B be the probability that all non trivial characteristics are fulfilled by at most B input pairs, i.e. $q_B = \Pr_{\substack{a,b \\ a \neq 0}}[N(a,b) \leqslant B]$. According to (Blondeau et al., 2013): $q_B \geqslant 1 - \frac{\Pi^B 2^{(n-1)B}}{(B+1)!} 2^{2n}$. Hence, for a fixed key K: if $q_B \cong 1$ then $N(a,b) \leqslant B$, and $\Pr_x[E_K(x) \oplus E_K(x \oplus a) = b] \leqslant \frac{B}{2^n}$.

White-Box Implementation of 2-$\mathsf{TEM}(\mathcal{P})$. We use the 8-bit instantiation of SPNbox (Bogdanov et al., 2016) as block cipher E. SPNbox-8 operates on $t = 16$ blocks of $m = 8$ bits and applies $R = 10$ rounds of the following transformations:

- A substitution γ: $\mathbb{F}_{2^8}^{16} \to \mathbb{F}_{2^8}^{16}$ applies a S-box S to each block as follows: $\gamma(X) = (S(X_0), \ldots, S(X_{15}))$.
- A linear transformation θ: $\mathbb{F}_{2^8}^{16} \to \mathbb{F}_{2^8}^{16}$ multiplies the state $(X_0, \ldots, X_{15})$ by a Maximum Distance Separable matrix as follows: $\theta(X) = (X_0, \ldots, X_{15}) \cdot M_8$ where:

$$M_8 = \mathsf{had}(\mathsf{0x08}, \mathsf{0x16}, \mathsf{0x8a}, \mathsf{0x01}, \mathsf{0x70}, \mathsf{0x8d}, \mathsf{0x24}, \mathsf{0x76}, \mathsf{0xa8}, \mathsf{0x91}, \mathsf{0xad}, \mathsf{0x48}, \mathsf{0x05}, \mathsf{0xb5}, \mathsf{0xaf}, \mathsf{0xf8})$$

  is an Hadamard-Cauchy matrix.
- A round-dependent transformation σ^r: $\mathbb{F}_{2^8}^{16} \to \mathbb{F}_{2^8}^{16}$ adds a round-dependent constant to the state as follows: $\sigma^r(X) = (X_0 \oplus C_0^r, \ldots, X_{15} \oplus C_{15}^r)$ for $C_i^r = (r-1) \cdot t + i + 1$.

For any key K the S-box of the substitution layer is computed as follows: for any 8-bit input X, encrypt X with a small-scale variant of the AES cipher using K as a master key and expanded using the SHAKE key derivation function. The variant of the AES cipher is composed of 64 rounds of the following transformations: SubBytes, AddRoundKey and MixColumns. The SubBytes transformation is composed of only one AES S-box and the Mixcolumns transformation applies the identity matrix to the state. If the S-box is precomputed and implemented by a lookup table then SPNbox-8 satisfies the unbreakability security.

Since $K = (K_1, K_2)$ and 2-$\mathsf{TEM}(\mathcal{P})$ needs four block cipher calls, the white-box implementation makes use of four pre-computed S-boxes with K_1, K_1^2, K_2, and K_2^2.

White-Box Security. The PWE satisfies the UBK-ATK security, for $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CCA}\}$ because of the unbreakability of SPNbox-8. Regarding the lockability security, it is guaranteed by the unforgeability and unclonability of the PUF instance. Hence, a tight estimation of the security bounds for a specific PUF class is necessary to get the security bounds for lockability.

3.4 Application

Our scheme is designed for exchange of encrypted data between a trusted server and a client on a mobile device. The server and the client share a secret key for the encryption: the client is given a white-box program with a fixed key. In addition, if a secure[2] PUF exists on the client's device, it is used to reinforce the security of the white-box program. Because of the uniqueness of a PUF instance, an enrollment phase is needed to "characterize" it. In other words, the trusted server stores some challenge-response pairs corresponding to the PUF thanks to an evaluation program SECURE.EVAL executed on the client device. Such an enrollment phase is used for PUF-based authentication. During the enrollment, some helper data are computed to enable the client to recover the enrolled responses from noisy ones. A helper data is computed by a SECURE.SKETCH procedure and stored by the server. It is a public side information that enables one to recover a string y from any noisy but close enough y'. Assume that the server sends some encrypted data to the client and the client needs to decrypt them. The server randomly picks a pair of challenge-response and encrypts the data using the shared key and the PUF response. Then the server sends the ciphertext together with the challenge and the helper data to the client. The client evaluates the PUF instance on the challenge and gets a noisy response. Thanks to a recovery procedure REC and the helper data, the client recovers the correct response and decrypts the ciphertext. We refer to (Dodis et al., 2008) for precise definitions of a secure sketch and a recovery procedure for noisy data.

REFERENCES

Armknecht, F., Moriyama, D., Sadeghi, A., and Yung, M. (2016). Towards a unified security model for physically unclonable functions. In *Topics in Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conference 2016*, volume 9610 of *Lecture Notes in Computer Science*, pages 271–287. Springer.

Biryukov, A., Bouillaguet, C., and Khovratovich, D. (2014). Cryptographic schemes based on the ASASA structure: Black-box, white-box, and public-key (extended abstract). In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security*, volume 8873 of *Lecture Notes in Computer Science*, pages 63–84. Springer.

Blondeau, C., Bogdanov, A., and Leander, G. (2013). Bounds in shallows and in miseries. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference*, volume 8042 of *Lecture Notes in Computer Science*, pages 204–221. Springer.

Bogdanov, A. and Isobe, T. (2015). White-box cryptography revisited: Space-hard ciphers. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1058–1069. ACM.

Bogdanov, A., Isobe, T., and Tischhauser, E. (2016). Towards practical whitebox cryptography: Optimizing efficiency and space hardness. In *Advances in Cryptology - ASIACRYPT 2016, Proceedings, Part I*, volume 10031 of *LNCS*, pages 126–158.

Cho, J., Choi, K. Y., Dinur, I., Dunkelman, O., Keller, N., Moon, D., and Veidberg, A. (2017). WEM: A new family of white-box block ciphers based on the even-mansour construction. In *Topics in Cryptology - CT-RSA 2017 - The Cryptographers' Track at the RSA Conference 2017*, volume 10159 of *Lecture Notes in Computer Science*, pages 293–308. Springer.

Chow, S., Eisen, P. A., Johnson, H., and van Oorschot, P. C. (2002a). White-box cryptography and an AES implementation. In *Selected Areas in Cryptography - SAC 2002*, volume 2595 of *Lecture Notes in Computer Science*, pages 250–270. Springer.

Chow, S., Eisen, P. A., Johnson, H., and van Oorschot, P. C. (2002b). A white-box DES implementation for DRM applications. In *Security and Privacy in Digital Rights Management, ACM CCS-9 Workshop, DRM 2002*, volume 2696 of *Lecture Notes in Computer Science*, pages 1–15. Springer.

Cogliati, B., Lampe, R., and Seurin, Y. (2015). Tweaking even-mansour ciphers. In *Advances in Cryptology - CRYPTO 2015*, volume 9215 of *LNCS*, pages 189–208. Springer Berlin Heidelberg.

Delerablée, C., Lepoint, T., Paillier, P., and Rivain, M. (2013). White-box security notions for symmetric encryption schemes. In *Selected Areas in Cryptography - SAC 2013*, volume 8282 of *Lecture Notes in Computer Science*, pages 247–264. Springer.

Dodis, Y., Ostrovsky, R., Reyzin, L., and Smith, A. D. (2008). Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139.

Fouque, P., Karpman, P., Kirchner, P., and Minaud, B. (2016). Efficient and provable white-box primitives. In *Advances in Cryptology - ASIACRYPT 2016 - Proceedings, Part I*, volume 10031 of *LNCS*, pages 159–188.

Liskov, M. D., Rivest, R. L., and Wagner, D. A. (2002). Tweakable block ciphers. In *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference*, volume 2442 of *Lecture Notes in Computer Science*, pages 31–46. Springer.

Maes, R. (2012). *Physically Unclonable Functions: Constructions, Properties and Applications*. PhD thesis, Katholieke Universiteit Leuven - Faculty of Engineering.

[2]Unpredictable and unclonable.