Towards a Theory of White-Box Security

Draft, full version

Amir Herzberg[*], Amitabh Saxena[†], Haya Shulman[‡], Bruno Crispo[§]

February 27, 2008

Abstract

Program hardening for secure execution in remote untrusted environment is an important yet elusive goal of security, with numerous attempts and efforts of the research community to produce secure solutions. Obfuscation is the prevailing practical technique employed to tackle this issue. Unfortunately, no provably secure obfuscation techniques currently exist. Moreover, Barak *et al.* in [1], showed that not all programs can be obfuscated.

We present a rigorous approach to *program hardening*, based on a new white box primitive, the *White Box Remote Program Execution (WBRPE)*, whose security specifications include confidentiality and integrity of both the local and the remote hosts. *WBRPE* can be used for many applications, e.g. grid computing, *Digital Rights Management*, mobile agents.

We construct a specific program such that if there exists a secure *WBRPE* for that program, then there is a secure *WBRPE* for *any* program, reducing its security to the underlying *WBRPE* primitive. This is the first proof by reduction among two white box primitives and it introduces new techniques that employ program manipulation.

1 Introduction

The goal of enabling secure execution of programs in remote untrusted environment is of high theoretical as well as practical importance, and is a basic requirement for many applications, such as *Digital Rights Management (DRM)*, grid computing, cooperative computing, e.g. file resource sharing (preventing free riders). This issue has received substantial attention during the last decade in the context of mobile code, along with proposals for solutions, targeted at addressing the problem in question by employing software or hardware based techniques, consequently producing a *Trusted Computing Base*, which essentially provides a secure haven on which the program is executed.

In hardware based approach, an additional hardware, that constitutes a secure trusted platform, is supplied, e.g. smartcards or trusted third parties, on which the secret data can be stored and the computations involving it performed. Hardware based approach produces solutions in black box security model, in which the attacker cannot access and observe the internals of the hardware, e.g. secret keys inside it, and can only control the input/ output behaviour of the system.

Applications that employ hardware benefit from high security promises. For instance, the deployment of smartcards in *Digital Rights Management (DRM)* solutions is extensive. *DRM* techniques enable software vendors and publishers to protect their intellectual property and to prevent software piracy, e.g. the smartcard is used to activate the licensing of a protected application. The secret is located inside the smartcard and never leaves it, such that all the computations are performed inside, consequently producing black box attack model. Various solutions based on the existence of a trusted third party were proposed, e.g. [2], where

[*] *Bar-Ilan University, Israel*
[†] *University of Trento, Italy*
[‡] *Bar-Ilan University, Israel, contact author: haya.shulman@gmail.com*
[§] *University of Trento, Italy*

they employ a trusted third party which performs computations on behalf of the mobile code but does not learn anything about the encrypted computation.

Although it seems that an additional hardware addresses the associated security issues in a best possible way providing provable security, there are disadvantages, which are inherent in high cost, unreliability and inflexibility of the hardware. In addition the security completely depends on the trust relationship with the additional hardware, e.g. trusted third party, thus making it inapplicable to many useful scenarios. Furthermore, in practice hardware alone is often not enough, since even hardware based solutions rely on software to accomplish the overall security, e.g. according to the *DRM* example above, the security of the intellectual property protected with a smartcard, relies on the security and the reliability of the software installed on the system, which is required to constantly probe the smartcard in order for the protected application to operate.

Therefore, we seek software only techniques for secure execution of programs in a remote untrusted environment. In addition to practical importance, understanding the level of security that can be attained by employing software only techniques is intriguing on its own, especially due to the belief that it is difficult to attain a reasonable level of security by employing software only approach, let alone a level of security comparable to black box security.

Specifically we investigate the *white-box security model*, which essentially means that once the program leaves the site of the originator, it is completely exposed to the attacker. Thus, in addition to the attacker's ability to controll the input/ output behaviour of the system, the code and data can be observed and manipulated. White box security employs software hardening techniques to withstand attacks in remote untrusted environments and aims to emulate the black box notion of security, against attackers which have full access to the software.

The natural question that emerges from the distinction between the two security models is "can we provide provable security in the white box security model?". A step towards obtaining provable security is to follow the cryptographic principles, initiating with the Kerkhoff principle: the secrecy should be in the keys but not in the algorithm. However, in white box model, as opposed to the black box, the attacker has access to the code and the data containing the secret keys. Can the software be hardened to hide the secret keys? The technique employed to harden the software should be known to the attacker, and secrecy has to rely on computational limitations of the attacker, like in black box security. To answer we recall the approaches employed to establish (provable) security in the *black-box model*:

- The unconditional proof of security, e.g. *One Time Pad*, is the best security one can hope for and holds for any attacker with infinite resources. However, unconditional security has limited applicability.
- Establishing security by failure to cryptanalyse. Practical building blocks are standardised by withstanding extensive cryptanalysis efforts, e.g. block ciphers [3] such as AES [4], DES [5].
- Proofs of security of one scheme by reduction to another scheme (building block).

To attain provable security, schemes that employ software hardening techniques should be based on *white-box building blocks*. Following the same idea as in black box, research efforts should be focused on standardizing practical building blocks by failure to cryptanalyze, and on constructing more advanced schemes on top of these building blocks, reducing the security of the schemes to the security of the (weaker) underlying building blocks.

Therefore, basing white box security on rigorous foundations, and the investigation of white box building blocks, is of theoretical and of practical importance and motivates our efforts to identify candidates for white box building blocks.

1.1 Obfuscation

Obfuscation is a candidate building block for white box security, which received substantial attention from theoreticians and practitioners. An obfuscator $\mathcal{O}$ is an efficient compiler that transforms a program P into a hardened program $\mathcal{O}(P)$, which pertains the functionality of the original program but is hard to analyse

and to reverse engineer. Obfuscation is the prevailing practical approach to software hardening, and was also investigated by theoreticians. However in both theory and practice, obfuscation exhibited insufficient results.

The impossibility result by Barak *et al.* [1] states that there does not exist a general obfuscator for any program. Although there are some positive results, e.g. Wee [6], these are restricted and do not suffice for practical applications.

In addition, practitioners cannot provide a unequivocal opinion regarding the security of obfuscators. Experts in practical obfuscation, e.g. Collberg [7], cannot say whether obfuscators can protect even simple programs, e.g. to hide intermediate state of modular programs, since an obfuscated program may leak information about the computation that it performs. A special case of obfuscation applied to hide the secret keys inside encryption programs is *White Box Cryptography (WBC)* [8, 9]. So far, proposed white box cryptography solutions were subsequently broken [10, 11]. For a discussion of related works on obfuscation see Appendix 7. The *WBRPE* scheme we present can be seen as an extension of white-box cryptography.

1.2 White Box Remote Program Execution (WBRPE)

Since existing obfuscators do not provide the required level of security, we need alternative *white-box security* building blocks. In this work, we propose the *White Box Remote Program Execution (WBRPE)* scheme, as a candidate for a white box building block. WBRPE can be employed to facilitate a variety of applications, e.g. grid computing and public online databases, and Digital Rights Management (DRM) applications.

In *Remote Program Execution*, programs are sent by a *local host* (a.k.a. the originator) for execution on a *remote host*, and possibly use some data available to the *remote host*; see Figure 1. The local and the remote hosts may not trust each other, and since the local host loses all control over the program, hence security issues need to be dealt with. In particular these include confidentiality and integrity of input programs supplied by the local host and confidentiality of inputs provided by the remote host.

We motivate the introduction of the *WBRPE* scheme by presenting two trivial solutions to this setting. One trivial solution, that provides confidentiality and integrity of the input programs, is to execute the program on the originator's site and have the remote host transfer the data to the local host. However, this imposes a heavy burden on the network, and increases the computation time to completion. In addition this does not protect the confidentiality of the remote input, e.g. a private database, such as [12].

The other trivial solution assumes that the remote host has an access to a black box, e.g. a smartcard. The local host will encrypt the program; the black box has the corresponding encryption/ decryption key pair and can therefore decrypt the program, compute the result of the program on the input supplied by the remote host, encrypt the result and return the response to the remote host, which forwards it to the local host. Furthermore, to prevent forgery of the input programs or of the result, the local host and the black box may employ digital signatures to verify the input programs and to sign the result of the computation. In addition, the black box can validate the input programs before execution, to prevent execution of invalid programs, e.g. that attempt to infer unauthorised or secret information about the remote input.

The *WBRPE* should satisfy similar requirements, employing software only techniques without assuming secure hardware, i.e. trusted third party or smartcards. The *WBRPE* scheme is composed of three efficient procedures, generation, hardening and unhardening, see Figure 1:

- The generation procedure produces two keys, (hardening and verification key) and a program, which we call the *obfuscated virtual machine* OVM.
- The hardening key is used by the hardening procedure to harden, e.g. encrypt and/ or authenticate, the input programs.
- The obfuscated virtual machine receives the hardened input program along with input from the remote host. It decodes the hardened program, e.g. decrypts and/ or validates it, and returns the result of the program applied to the inputs. The result is encoded, e.g. encrypted and/or authenticated (within the OVM).

- The unhardening procedure unhardens, e.g. decrypts and validates the result received from the remote host.

In most of this paper we assume that the hardening key is a shared secret between the remote host and a particular local host. In some applications it is convenient to use a public hardening key, e.g. allowing all users to send the agents to the remote server; this extension is presented in Section 2.4.

1.3 White Box RPE for ALL Programs

The negative result by Barak *et al.* [1], shows that an obfuscator for all programs does not exist, however this result does not imply that there cannot be alternative hardening schemes which would work for any program. In particular, is there a *WBRPE* for all programs?

To address this question we present a specific program, denoted UP (for *universal program*), with parameter K (key). Given a *WBRPE* scheme that works for the family of universal programs $\{\mathrm{UP}_K\}$, we present a *'Universal' WBRPE* scheme that works for any program, i.e. provides the security specifications of *WBRPE* for *any* input program.

We establish the security of the *Universal WBRPE* scheme by reducing to the security of the underlying *WBRPE* scheme. A reduction is a basic technique in cryptography; we believe this to be the first proof by reduction between two white box primitives. White box reductions differ in several aspects from the classical cryptographic black box reductions, most notably in the way we construct and represent programs as strings and introduce code manipulation techniques. These reduction techniques can be applicable in other works in white box security.

Organisation

The rest of this work is organised as follows. In section 2 we introduce the definition of the *WBRPE* scheme along with the security specifications. In section 3 we present a *Universal WBRPE* construction, and prove its security for each security specification defined for *WBRPE* scheme in Section 2. In Section 5, we present a construction of the *Universal WBRRPEwV* that employs program validation to provide privacy of the remote inputs, and prove its security. We then review several applications of *WBRPE* 6. In Section 7 we review related works and Section 8 provides concluding remarks and open questions that are yet to be addressed.

2 White-Box RPE Definitions

A *WBRPE* scheme W is comprised of three efficient algorithms, $(\mathcal{G},\mathcal{H},\mathcal{U})$ for generation, hardening and unhardening, respectively. The generation procedure $\mathcal{G}$ generates the obfuscated virtual machine *OVM*, the hardening key hk and the verification key vk. The hardening procedure applied on some input program, computes the hardened program, e.g. encryption and/ or authentication of the original program, and produces two outputs, the hardened program and a one time unhardening key. The remote host passes the hardened program, along with the remote input a to the OVM for execution. The OVM has the required keys, and can therefore extract and evaluate the program. Next, the OVM computes the result of P on a, and returns the (hardened) output. The local host, upon receipt the hardened output, applies the unhardening procedure with the unhardening key, to obtain the final result of the computation.

Given a turing machine $P \in \mathbb{TM}$, let $P(a)$ denote a value of the computation of P on a. We introduce a time parameter, to hide the time that it takes each program to execute, and the length parameter to hide the length of the result. Let $P_{t,l}(a) = P_t(a)[1...l]$ denote an l bit value of the t step computation of P on input a. The definition follows.

Definition 2.1 *A* White Box RPE (WBRPE) *scheme W for programs family $P_{kk\in\mathbb{N}}$ consists of a tuple $W = \langle\mathcal{G},\mathcal{H},\mathcal{U}\rangle$ of PPT algorithms satisfying the following conditions:*

For all $(hk, vk, OVM) \xleftarrow{R} \mathcal{G}(1^k), a \in \{0,1\}^*, P \in \mathbb{TM}$, $t, l \in \mathbb{N}$ *and* $(c, uk) \leftarrow \mathcal{H}_{hk}(P)$, *holds:*

- $OVM \in PPT$
- $P_{t,l}(a) = \mathcal{U}_{uk,vk}(OVM(c, a, t, l))$

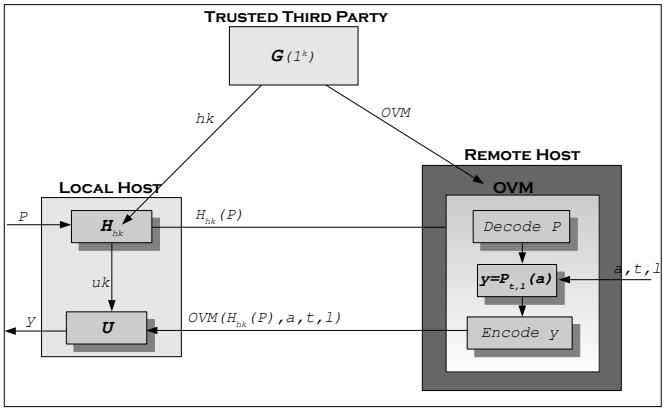

Figure 1: *WBRPE* scheme.

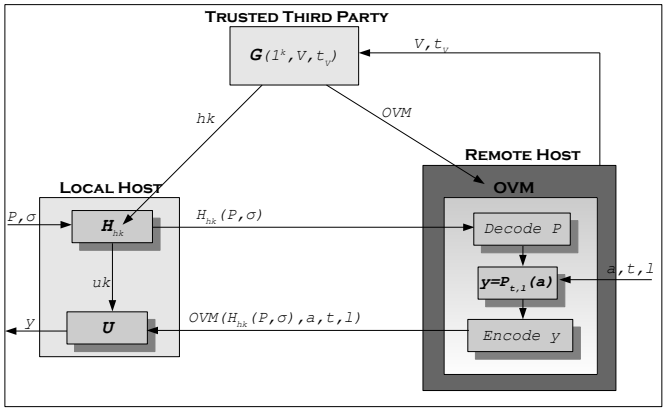

Figure 2: *WBRPE* scheme with privacy.

2.1 Indistinguishability of the Local Inputs Specification

The first security specification we consider is to hide the contents of the input programs from the remote host. To ensure local inputs privacy we employ a variation of the indistinguishability experiment for encryption schemes [13]. We specify the indistinguishability definition w.r.t. a PPT algorithm $A = (A_1, A_2)$, denoting by $\mathcal{HO}$ the hardening oracle which the algorithm A obtains access to, during the indistinguishability experiment. The experiment is described in detail in Figure 3 we now give an informal definition.

As its first step the experiment generates the keys and the obfuscated virtual machine. Next it invokes the adversarial algorithm and provides it with an oracle access to the hardening functionality for its hardening queries, passes it the obfuscated virtual machine and the public verification key. Each application of the hardening procedure generates a hardened program and a one time unhardening key. Eventually the adversary outputs two programs of equal size. The experiment tosses a bit b and one of the programs is subsequently hardened. During the second phase the adversary keeps an oracle access to $\mathcal{HO}$, obtains the hardened challenge program and has to distinguish. If the adversary guesses correctly, the experiment returns 1, i.e. the adversary won, and otherwise returns 0, the adversary lost.

In the Definition 2.2 the experiment obtains in an input a sequence of random coins sequence r, which it splits between the random coins used by the generation procedure $\mathcal{G}$, the pair of PPT algorithms $A = (A_1, A_2)$, the bit b chosen by the experiment and the hardening procedure applied by the experiment to generate the challenge. Therefore given the same r the execution of the experiment is a deterministic function of the inputs.

Definition 2.2 *(Indistinguishability) Let* $W = (\mathcal{G}, \mathcal{H}, \mathcal{U})$ *be a* WBRPE *scheme and let* $A = (A_1, A_2)$ *be a pair of PPT algorithms. For* $k \in \mathbb{N}$, $r \in \{0,1\}^*$ *we define the advantage of the adversary* A *in the* $WB - IND - CPA$ *experiment as follows:*

$$\boldsymbol{Adv}_{W,A}^{WB-IND-CPA}(k) = 2 * \Pr[\boldsymbol{Exp}_{W,A}^{WB-IND-CPA}(k; r) = 1] - 1$$

Where the probabilities are taken over r *and the experiment* $\boldsymbol{Exp}_{W,A}^{WB-IND-CPA}(k; r)$ *is defined in Figure 3. A* WBRPE *scheme* W *is* $WB - IND - CPA$ *secure if the advantage function* $\boldsymbol{Adv}_{W,A}^{WB-IND-CPA}(\cdot)$ *is negligible over all PPT adversarial algorithms* A.

$\mathtt{Exp}_{W,A}^{WB-IND-CPA}(k;r)\{$

$(r_{\mathcal{G}}||r_{\mathcal{HO}}||r_{A_1}||r_b||r_{\mathcal{H}}||r_{A_2}) \leftarrow r$

$\langle hk, vk, \mathrm{OVM}\rangle \leftarrow \mathcal{G}(1^k; r_{\mathcal{G}})$

$(P_0, P_1, s) \leftarrow A_1^{\mathcal{HO}(\cdot,hk;r_{\mathcal{HO}})}(1^k, \mathrm{OVM}, vk, r_{A_1})$

$b \leftarrow r_b$

$(c_b, uk_b) \leftarrow \mathcal{H}_{hk}(P_b; r_{\mathcal{H}})$

$b' \leftarrow A_2^{\mathcal{HO}(\cdot,hk;r_{\mathcal{HO}})}(c_b, s; r_{A_2})$

if $((b = b') \ \wedge \ (|P_0| = |P_1|))\{$

return 1

}

return 0

}

Figure 3: *WBRPE* indistinguishability experiment. Where $\mathcal{HO}(P, hk) = \mathcal{H}_{hk}(P)$ is the hardening oracle.

$\mathtt{Exp}_{W,A}^{WB-UNF-OUT}(k;r)\{$

$(r_{\mathcal{G}}||r_{\mathcal{HO}}||r_A) \leftarrow r$

$\langle hk, vk, \mathrm{OVM}\rangle \leftarrow \mathcal{G}(1^k; r_{\mathcal{G}})$

$(\omega, P, t, uk) \leftarrow A^{\mathcal{HO}(\cdot,hk;r_{\mathcal{HO}})}(1^k, \mathrm{OVM}, vk; r_A)$

$y \leftarrow \mathcal{U}_{uk,vk}(\omega, P, t)$

if $(y = \perp)\{$

return 0

}

if $(\forall a \in \{0,1\}^*, y \neq P_{t,|y|}(a))\{$

return 1

}

return 0

}

Figure 4: *WBRPE* output unforgeability experiment.

2.2 Unforgeability Specification

In typical scenarios, e.g. shopping mobile agent, the adversary may try to change the programs sent by the originator to programs of his choice, such that instead of looking for the best offer the agent purchases the most expensive item. Further, the adversary may try to change the result of the computation to some other result. Our goal is to circumvent adversarial attempts to forge the result output by the scheme, and this is achieved by the unforgeability specification.

We extend the definition of *WBRPE* scheme, such that the unhardening procedure $\mathcal{U}$ can obtain additional optional parameters in an input, when validation of the inputs is required. More specifically, the local host can validate the result, i.e. the result of the computation is indeed of the input program P that it supplied for the specified number of steps t. To perform validation, the unhardening procedure $\mathcal{U}$ will use the public verification key vk. The implication is that everyone can validate the result, but one the possessor of the secret unhardening key uk can obtain the final result. The validation of the result is optional and can only be performed when P and t are supplied in addition to ω, i.e. the output of OVM. The validation is required if the result of the program that the originator sends for execution is forwarded to some other recipient. To verify the integrity of the result the recipient can apply the unhardening procedure, on ω, an input program P, and the t that was used for programs execution. The unhardening procedure uses the verification key vk, that is public and the unhardening key uk, which can be transfered securely by the originator.
We consider two types of forgery of *WBRPE* scheme W: in particular, we introduce the notion of output forgery, i.e. the result of the computation is an incorrect output and could not have been generated by the input program on any remote input, and of program forgery. Below, we give an intuitive description of the unforgeability experiment, followed by the presentation of both types of forgery.

The experiment applies the generation procedure and obtains hardening key, verification key and OVM. It then invokes the adversary with an oracle access to hardening functionality and the OVM and the verification key in an input. Eventually the adversary outputs the hardened result of the computation ω, input program P, the number of steps t and the unhardening key uk. The experiment applies the unhardening procedure $\mathcal{U}$ on ω, P and t, and obtains the result of the computation y. If y is valid the experiment checks if it is a forgery and if yes, returns 1, i.e. the adversary successfully generated a forgery, otherwise returns 0, the adversary failed.

2.2.1 Output Forgery

Consider a public online database, where a user queries the database with some query, and the adversary changes the result of the computation to some other value of his choice. We prevent this using the "correct output" specification.

This type of forgery means that the output is not a result of the computation of the input program on any remote input. Specifically, the adversary successfully generated an output tuple (ω, P, t, uk), s.t. the result $y \leftarrow \mathcal{U}_{uk,vk}(\omega)$ could not have been generated by the hardened program P, i.e. $\forall a\ y \neq P_{t,|y|}(a)$.

Definition 2.3 *(Unforgeability - Correct Output) Let $W = (\mathcal{G}, \mathcal{H}, \mathcal{U})$ be a WBRPE scheme and let A be a PPT algorithm. For $k \in \mathbb{N}$ we define the advantage of the adversary A in the unforgeability experiment as follows:*

$$\boldsymbol{Adv}_{W,A}^{WB-UNF-OUT}(k) = \Pr[\boldsymbol{Exp}_{W,A}^{WB-UNF-OUT}(k) = 1]$$

Where $\boldsymbol{Exp}_{W,A}^{WB-UNF-OUT}(k)$ as defined in Figure 4. A WBRPE *scheme W is $WB-UNF-OUT$ secure, if the advantage $\boldsymbol{Adv}_{W,A}^{WB-UNF-OUT}(\cdot)$ is a negligible function for all PPT adversarial algorithms A.*

2.2.2 Program Forgery

In this type of forgery, the legitimate party never queried the hardening oracle with a program for which the result was generated. Instead, the adversary replaces the authentic hardened program with some other program (replay or a forgery). We consider two variants:

- The adversary successfully generated a new unhardening key uk, which was not output by the hardening oracle. Namely, it generated a tuple (ω, P, t, uk), s.t. $y \leftarrow \mathcal{U}_{uk,vk}(\omega)$ and $y = P_{t,|y|}(a)$ for some $a,\ t,\ l$.
- The adversary successfully generated an output (ω, P, t, uk) s.t. $y \leftarrow \mathcal{U}_{uk,vk}(\omega)$, and $y = P_{t,|y|}(a)$, where the unhardening key uk was generated for a different program P'. In both cases, the adversary did not perform a hardening oracle query on P.

Let UK denote the set of second argument (uk) of $\mathcal{HO}$ responses and let $P[uk]$ be an array containing the queries submitted by the adversary to the hardening oracle during the experiment, indexed by a corresponding key uk generated by the oracle as a reply.

Definition 2.4 *(Unforgeability - Correct Program) Let $W = (\mathcal{G}, \mathcal{H}, \mathcal{U})$ be a WBRPE scheme and let A be a PPT algorithm. For $k \in \mathbb{N}$, $r \in \{0,1\}^*$ we define the advantage of the adversary A in the unforgeability experiment as follows:*

$$\boldsymbol{Adv}_{W,A}^{WB-UNF-PRG}(k) = \Pr[\boldsymbol{Exp}_{W,A}^{WB-UNF-PRG}(k; r) = 1]$$

Where $\boldsymbol{Exp}_{W,A}^{WB-UNF-PRG}(k; r)$ is defined in Figure 5. A WBRPE *scheme W is $WB-UNF-PRG$ secure, i.e. provides output integrity, if the advantage $\boldsymbol{Adv}_{W,A}^{WB-UNF-PRG}(\cdot)$ is a negligible function for all PPT adversarial algorithms A.*

2.3 White Box RPE with Validation (WBRPEwV)

In various scenarios, e.g. when the remote input is a database that contains private medical or personal information, it is necessary to limit the information about the remote input that the local host may obtain. To address this, we allow the owner of the remote input to specify the set of valid queries on the database during the generation phase of the scheme, thus the obfuscated virtual machine is confined to executing valid programs only. Defining the valid queries set, also prevents manipulation of the database by the adversary, i.e. deleting or modifying entries.

$\mathtt{Exp}_{W,A}^{WB-UNF-PRG}(k;r)\{$

$(r_{\mathcal{G}}||r_{\mathcal{HO}}||r_A) \leftarrow r$

$UK = \emptyset$

$\langle hk, vk, \mathrm{OVM}\rangle \leftarrow \mathcal{G}(1^k; r_{\mathcal{G}})$

$(\omega, P, t, uk) \leftarrow A^{\mathcal{HO}(\cdot,hk;r_{\mathcal{HO}})}(1^k, \mathrm{OVM}, vk; r_A)$

$y \leftarrow \mathcal{U}_{uk,vk}(\omega, P, t)$

if $(y \neq \perp)\{$

return 0

$\}$

if $((uk \notin UK) \vee (P \notin P[uk]))\{$

return 1

$\}$

return 0

$\}$

Figure 5: *WBRPE* program unforgeability specification.

Oracle $\mathcal{HO}(P, hk)\{$

$\langle c, uk\rangle \leftarrow \mathcal{H}_{hk}(P)$

$P[uk] \leftarrow P$

$UK = UK \cup \{uk\}$

return $\langle c, uk\rangle$

$\}$

Figure 6: The hardening oracle $\mathcal{HO}$ used in $WB-UNF-PRG$ experiment.

To restrict the execution to valid programs, we introduce the following supplementary parameters to the definition 2.1 of *WBRPE*: the validation program $V \in \mathbb{TM}$ and the number of steps t_V to execute V, which are given to the trusted party that performs the generation phase. On input a program $P \in \mathbb{TM}$ and a validation parameter $\sigma \in \{0,1\}^*$, $V_{t_V,1}(P, \sigma) \in \{0,1\}$ returns 1 if the program is valid and 0 otherwise. The OVM will only execute valid programs. In addition, the signature of the hardening procedure $\mathcal{H}$ is modified and along with the input program P, it will also receive the validation parameter σ. The definition below:

Definition 2.5 *(*WBRPE *with Validation) A* white box RPE with Validation (WBRPEwV) *scheme W, consists of a tuple $\langle \mathcal{G}, \mathcal{H}, \mathcal{U}\rangle$ of PPT algorithms satisfying the following conditions:*

For all $\langle hk, vk, OVM\rangle \leftarrow \mathcal{G}(1^k, V, t_V)$, s.t. $V \in \mathbb{TM}, t_V \in \mathbb{N}$, $a \in \{0,1\}^*, P \in \mathbb{TM}, \sigma \in \{0,1\}^*$, $t, l \in \mathbb{N}$, and $(c, uk) \leftarrow \mathcal{H}_{hk}(P, \sigma)$, holds:

- *$OVM \in PPT$*
- *if $(V_{t_V,1}(P, \sigma) = 1)$ then $P_{t,l}(a) = \mathcal{U}_{uk,vk}(OVM(c, a, t, l))$*

The *WBRPEwV* scheme provides the indistinguishability and the unforgeability specifications that were defined for the *WBRPE* scheme, and also the privacy of remote inputs specification.

2.3.1 Privacy of Remote Inputs Specification

In the definition of the privacy specification below we require that the adversary gains no additional information about the remote input than what is already a-priori known to it, given the result of the computation and access to the scheme. We formalize this using the simulation paradigm: the *WBRPE* scheme W is semantically secure w.r.t. privacy of remote inputs, if everything an adversary can learn about the remote input given an access to the scheme for the computation of the result, the simulator could learn without any access to the scheme, by resorting to the help of a trusted third party to execute and validate programs on its behalf.
In the definition below the function f is the information about the remote input that the adversary attempts

to learn and the function h represents the adversary's a-priori knowledge regarding the remote input. We denote by $\{REM_k\}_{k\in\mathbb{N}}$ the probability ensemble representing the distribution of the remote inputs and there exists a single polynomial $p(\cdot)$ such that for all sufficiently large k's, $|REM_k| \leq p(k)$, which essentially implies that there exists a polynomial bound on the length of the strings in this distribution. The adversary's inability to learn information about the remote input should hold for any distribution of remote inputs.

In the definition of privacy of remote inputs we present two environments, the white box environment which emulates the the real execution, and the black box environment that emulates the ideal execution, such that the black box environment provides no access to the scheme but simulates a trusted third third party which carries out the computations and the validations of the input programs on behalf of the local host, whereas the white box environment simulates an adversary which exploits the scheme to gain some additional information about the remote input. The advantage that the adversary gains in the white box, i.e. real, environment should be almost the same, as the advantage that the simulator gains in the black box, i.e. ideal, environment.
The adversary may have, a possibly limited, control over the remote input, e.g. consider a scenario where the remote input supplied by the executing remote host is a database. Clearly, the adversary may insert or update entries in the database. We model this by introducing the a_{ADV} parameter, which is the part of the remote input that is under the control of the adversary.

Definition 2.6 *(Privacy of Remote Inputs) Let $WV = (\mathcal{GV}, \mathcal{HV}, \mathcal{UV})$ be a WBRPEwV scheme. Let $A = (A_1, A_2, A_3)$ and $S = (S_1, S_2)$ be* PPT *algorithms, let $\{REM_k\}_{k\in\mathbb{N}}$ be a polynomial time sampleable random variable and let $h : 1^k \times REM_k \to \{0,1\}^*$ be a polynomial time computable function and $f : 1^k \times REM_k \to \{0,1\}^*$ some function and $p(\cdot)$ positive polynomial. For $k \in \mathbb{N}$ we define the advantage of the adversary in the privacy of remote inputs experiment as follows:*

$$\boldsymbol{Adv}^{WB-PRV}_{WV,A}(k) = \Big| \Pr[\boldsymbol{Exp}^{WB-PRV}_{WV,A}(k, REM_k, h) = f(1^{|REM_k|}, REM_k)] -$$

$$- \Pr[\boldsymbol{Exp}^{BB-PRV}_{A_1,S}(k, REM_k, h) = f(1^{|REM_k|}, REM_k)] \Big|$$

Where $\boldsymbol{Exp}^{WB-PRV}_{WV,A}(\cdot)$ and $\boldsymbol{Exp}^{BB-PRV}_{A_1,S}(\cdot)$ are defined in Figures 7 and 8. A WBRPEwV *scheme WV is $WB-PRV$ secure if the advantage function $\boldsymbol{Adv}^{WB-PRV}_{WV,A}(\cdot)$ is negligible for all PPT adversarial algorithms A, every polynomial time sampleable random variable $\{REM_k\}_{k\in\mathbb{N}}$, every polynomial time computable function $h : 1^k \times REM_k \to \{0,1\}^*$ and every function $f : 1^k \times REM_k \to \{0,1\}^*$.*

2.3.2 Indistinguishability of Local Inputs with Validation Specification

This definition is similar in nature to the Definition 2.2, with the following modifications, the algorithm A_1 along with two challenge programs outputs two validation parameters for each program, s.t. the validation parameters are equal in length. The experiment tosses a bit and the corresponding input program along with the validation parameter are hardened by the experiment and are given to the algorithm A_2 in an input, which has to distinguish which of the challenge pair (program and validation parameters) was hardened. The formal definition is presented below, and for completeness we also present the indistinguishability with privacy experiment 9, which differs from the indistinguishability experiment defined in 3 in the validation parameter σ.

Definition 2.7 *(Indistinguishability with Validation) Let $WV = (\mathcal{GV}, \mathcal{HV}, \mathcal{UV})$ be a* WBRPEwV *scheme and let $A = (A_1, A_2)$ be a pair of PPT algorithms. For $k \in \mathbb{N}$ we define the advantage of the adversary A in the $WB-IND-CPA$ experiment as follows:*

$$\boldsymbol{Adv}^{WB-IND-CPA}_{WV,A}(k) = 2 * \Pr[\boldsymbol{Exp}^{WB-IND-CPA}_{WV,A}(k) = 1] - 1$$

Where the probabilities are taken over the coin tosses of A, the scheme WV and the experiment $\boldsymbol{Exp}^{WB-IND-CPA}_{WV,A}(k)$ as defined in Figure 9. A WBRPE *scheme WV is $WB-IND-CPA$ secure if the advantage function $\boldsymbol{Adv}^{WB-IND-CPA}_{WV,A}(\cdot)$ is negligible over all PPT adversarial algorithms A.*

$\mathtt{Exp}^{WB-PRV}_{WV,A}(k, REM_k, H)\{$

$a \xleftarrow{R} REM_k$
$\tau \leftarrow H(1^{|a|}, a)$
$(V, t_V, s) \leftarrow A_1^{REM_k}(1^k, \tau)$
$(hk, vk, \text{OVM}) \leftarrow \mathcal{GV}(1^k, V, t_V)$
$(c, a_{ADV}, t, l, uk, s) \leftarrow A_2^{REM_k}(\text{OVM}, hk, vk, s)$
$a \leftarrow (a||a_{ADV})$
$\omega \leftarrow \text{OVM}(c, a, t, l)$
return $A_3^{REM_k}(\omega, uk, s)$

$\}$

Figure 7: *WBRPE* $WB - PRV$ privacy experiment.

$\mathtt{Exp}^{BB-PRV}_{A_1,B}(k, REM_k, H)\{$

$a \xleftarrow{R} REM_k$
$\tau \leftarrow H(1^{|a|}, a)$
$(V, t_V, s) \leftarrow A_1^{REM_k}(1^k, \tau)$
$(P, \sigma, a_{ADV}, t, l, s) \leftarrow S_1^{REM_k}(V, t_V, \tau, s)$
$a \leftarrow (a||a_{ADV})$
if $(V_{t_V,1}(P, \sigma))\{$
$y \leftarrow P_{t,l}(a)$
$\}$
else $\{$
$y \leftarrow \perp$
$\}$
return $S_2(y, s)$

$\}$

Figure 8: *WBRPEwV* $BB - PRV$ privacy experiment.

2.3.3 Unforgeability with Validation Specification

The definitions of output and program unforgeability specifications of the *WBRPEwV* scheme are similar to the ones defined for *WBRPE* scheme. Further, since the validation parameter σ is only of importance when input to OVM along with the program P, it has no effect neither on the output nor on the program forgeries. Therefore the unforgeability experiments are identical to the ones defined for *WBRPE* scheme, in Figures 4, 5. The only difference is introduced in the definition of the hardening oracle of the program forgery experiment, where an adversary can forge the pair (P, σ). Intuitively, if the adversary replaces (P, σ) with an invalid P for σ then the validation will fail. Therefore as long as the supplied σ is valid, we do not consider this a forgery. Therefore, both the correct output and the correct program requirements hold for *WBRPEwV* definition.

Definition 2.8 *(Unforgeability with Validation - Correct Output) Let* $WV = (\mathcal{GV}, \mathcal{HV}, \mathcal{UV})$ *be a* WBRPEwV *scheme and let* A *be a PPT algorithm. For* $k \in \mathbb{N}$ *we define the advantage of the adversary* A *in the output unforgeability with validity experiment similarly to definition 2.3 and the experiment* $\mathtt{Exp}^{WB-UNF-OUT}_{WV,A}(k)$ *is defined analogously to the unforgeability experiment of the* WBRPE *scheme in Figure 4.*

Definition 2.9 *(Unforgeability with Validation - Correct Program) Let* $WV = (\mathcal{GV}, \mathcal{HV}, \mathcal{UV})$ *be a* WBRPEwV *scheme and let* A *be a PPT algorithm. For* $k \in \mathbb{N}$, $r \in \{0,1\}^*$ *we define the advantage of the adversary* A *in the program unforgeability with validity experiment similarly to definition 2.4 and the experiment* $\mathtt{Exp}^{WB-UNF-OUT}_{WV,A}(k)$ *is defined analogously to the unforgeability experiment of the* WBRPE *scheme in Figure 5.*

2.4 Asymmetric White Box RPE

In the security specifications presented in section 2 we focused on the symmetric *WBRPE* in which, there was a shared key hk between the OVM and the local host. This hk key is secret and is employed by the local host to harden programs and by the OVM to subsequently unharden them for execution. This implies that there is a unique OVM for every local host. In this section we introduce the *WBRPE* which supports the

$\mathtt{Exp}^{WB-IND-CPA}_{WV,A}(k)\{$

$\langle hk, vk, \mathrm{OVM}\rangle \leftarrow \mathcal{GV}(1^k)$

$(P_0, \sigma_0, P_1, \sigma_1, s) \leftarrow A_1^{\mathcal{HVO}(\cdot,hk)}(1^k, \mathrm{OVM}, vk)$

$b \in_R \{0,1\}$

$(c_b, uk_b) \leftarrow \mathcal{HV}_{hk}(P_b, \sigma_b)$

$b' \leftarrow A_2^{\mathcal{HVO}(\cdot,hk)}(c_b, s)$

if $((b = b') \ \wedge \ (|P_0| = |P_1|))\{$

return 1

$\}$

return 0

$\}$

Figure 9: *WBRPEwV* indistinguishability experiment; $\mathcal{HVO}(P, \sigma, hk) = \mathcal{HV}_{hk}(P, \sigma)$ is the hardening oracle.

extended definition where the hardening key is public. The definition of the signature of *WBRPE* remains the same, and the modification is introduced in the definitions of security specifications, where the attacking algorithm obtains the public hardening key hk in an input. In the asymmetric *WBRPE* the hardening key hk is public, and there is a corresponding unhardening key embedded inside the OVM. Namely, everyone can harden programs and send to OVM for execution, and only the OVM can unharden the programs, which implies the asymmetry. The obvious advantage of the asymmetric *WBRPE* is in its flexibility, i.e. new hosts can join the system without any effort, e.g. a marketplace scenario where everyone can work with one central remote host and the same OVM.

In the sequel we introduce a flag $\varphi \in \{PK, SK\}$, and when $\varphi = PK$ we refer to an asymmetric *WBRPE*, while $\varphi = SK$ denotes a symmetric *WBRPE*. This introduces several modifications in the implementation of indistinguishability and unforgeability security specifications. More specifically when $\varphi = PK$ the adversary receives the public hardening key hk in an input and can harden the programs by itself.

Definition 2.10 *(Indistinguishability) Let $W = (\mathcal{G}, \mathcal{H}, \mathcal{U})$ be a WBRPE scheme and let $A = (A_1, A_2)$ be a pair of PPT algorithms. For $\varphi \in \{PK, SK\}$ and for $k \in \mathbb{N}$ we define the advantage of the adversary A in the $WB - IND - CPA - \varphi$ experiment as follows*

$$\boldsymbol{Adv}^{WB-IND-CPA-\varphi}_{W,A}(k) = 2 * \Pr[\boldsymbol{Exp}^{WB-IND-CPA-\varphi}_{W,A}(k) = 1] - 1$$

Where the probabilities are taken over the coin tosses of A, the scheme W and the experiment $\boldsymbol{Exp}^{WB-IND-CPA-\varphi}_{W,A}(k)$ is defined in Figure 10. A WBRPE scheme W is $WB - IND - CPA - \varphi$ secure if the advantage function $\boldsymbol{Adv}^{WB-IND-CPA-\varphi}_{W,A}(\cdot)$ is negligible over all PPT adversarial algorithms A.

Definition 2.11 *(Unforgeability - Correct Output) Let $W = (\mathcal{G}, \mathcal{H}, \mathcal{U})$ be a WBRPE scheme and let A be a PPT algorithm. For $\varphi \in \{PK, SK\}$ and for $k \in \mathbb{N}$ we define the advantage of the adversary A in the unforgeability experiment as follows*

$$\boldsymbol{Adv}^{WB-UNF-OUT-\varphi}_{W,A}(k) = \Pr[\boldsymbol{Exp}^{WB-UNF-OUT-\varphi}_{W,A}(k) = 1]$$

Where $\boldsymbol{Exp}^{WB-UNF-OUT-\varphi}_{W,A}(k)$ is defined in Figure 12. A WBRPE scheme W is $WB - UNF - OUT - \varphi$ secure, i.e. provides input programs integrity, if the advantage $\boldsymbol{Adv}^{WB-UNF-OUT-\varphi}_{W,A}(\cdot)$ is a negligible function for all PPT adversarial algorithms A.

$\mathbf{Exp}_{W,A}^{WB-IND-CPA-\varphi}(k)$**:**

$\langle hk, vk, \text{OVM}\rangle \leftarrow \mathcal{G}(1^k)$

$(P_0, P_1, s) \leftarrow A_1^{\mathcal{HO}(\cdot,hk,\varphi)}(1^k, \text{OVM}, vk)$

$b \in_R \{0,1\}$

$(c_b, uk_b) \leftarrow \mathcal{H}_{hk}(P_b)$

$b' \leftarrow A_2^{\mathcal{HO}(\cdot,hk,\varphi)}(c_b, s)$

If $((b = b') \wedge (|P_0| = |P_1|))$ return 1 else return 0

Figure 10: The *WBRPE* indistinguishability experiment for φ.

Oracle $\mathcal{HO}(P, hk)$

if $(\varphi = PK)$

return (hk)

else

return $(\mathcal{H}_{hk}(P))$

Figure 11: The implementation of $\mathcal{HO}$.

$\mathbf{Exp}_{W,A}^{WB-UNF-OUT-\varphi}(k)$

$\langle hk, vk, \text{OVM}\rangle \leftarrow \mathcal{G}(1^k)$

$(\omega, P, t, uk) \leftarrow A^{\mathcal{HO}(\cdot,hk,\varphi)}(1^k, \text{OVM}, vk)$

$y \leftarrow \mathcal{U}_{uk,vk}(\omega, P, t)$

if $((y \neq \perp) \wedge (\forall a \in \{0,1\}^*,\ y \neq P_{t,|y|}(a)))$

return 1

return 0

Figure 12: The *WBRPE* output unforgeability experiment. Relevant for symmetric and asymmetric cases.

Let UK denote the set of second argument (uk) of $\mathcal{HO}$ responses and let $P[uk]$ be an array containing the queries submitted by the adversary to the hardening oracle during the experiment, indexed by a corresponding key uk generated by the oracle as a reply.

Definition 2.12 *(Unforgeability - Correct Program) Let $W = (\mathcal{G}, \mathcal{H}, \mathcal{U})$ be a WBRPE scheme and let A be a PPT algorithm. For $k \in \mathbb{N}$ we define the advantage of the adversary A in the unforgeability experiment as follows*

$$\boldsymbol{Adv}_{W,A}^{WB-UNF-PRG}(k) = \Pr[\boldsymbol{Exp}_{W,A}^{WB-UNF-PRG}(k) = 1]$$

Where $\boldsymbol{Exp}_{W,A}^{WB-UNF-PRG}(k)$ is defined in Figure 13. A WBRPE scheme W is $WB-UNF-PRG$ secure, i.e. provides output integrity, if the advantage $\boldsymbol{Adv}_{W,A}^{WB-UNF-PRG}(\cdot)$ is a negligible function for all PPT adversarial algorithms A.

The definition for privacy of remote inputs security specification trivially applies to public hardening key setting, and is presented in Definition 2.6.

3 Universal WBRPE

In this section we show that if there exists a *WBRPE* scheme that satisfies the security specifications for a *specific* family of universal programs, UP then there exists a *Universal WBRPE* scheme that satisfies the security specifications for every program.

$\mathtt{Exp}^{WB-UNF-PRG}_{W,A}(k)$

$UK = \emptyset$

$\langle hk, vk, \mathrm{OVM}\rangle \leftarrow \mathcal{G}(1^k)$

$(\omega, P, t, uk) \leftarrow A^{\mathcal{HO}(\cdot,hk)}(1^k, \mathrm{OVM}, vk)$

$y \leftarrow \mathcal{U}_{uk,vk}(\omega, P, t)$

if $((y \neq \perp) \ \wedge \ ((uk \notin UK) \ \vee \ (P \notin P[uk])))$

return 1

return 0

Figure 13: The *WBRPE* program unforgeability experiment. Relevant for symmetric case only (the adversary obtains an oracle access to the hardening functionality)

Oracle $\mathcal{HO}(P, hk)$

$\langle c, uk\rangle \leftarrow \mathcal{H}_{hk}(P)$

$P[uk] \leftarrow P$

$UK = UK \cup \{uk\}$

return $\langle c, uk\rangle$

Figure 14: The hardening oracle $\mathcal{HO}$.

The Universal Program UP

Let $\Pi = (\mathcal{G}_{AE}, \mathcal{AE}, \mathcal{VD})$ be an authenticated encryption scheme, that performs encryption and authentication, and decryption and validation of inputs, see Bellare and Rogaway [14]. The universal program UP_K (in Figure 15) is a Turing machine, that is created and instantiated with a secret key K, by the hardening procedure $\mathcal{H}$. When invoked by the obfuscated virtual machine OVM, the universal program UP_K reads a' of the input tape, and parses it to obtain (a, t, l, c_P), i.e. the remote input, the number of steps of program's execution, the length of the output and the encrypted program. UP decrypts and validates c_P using the key K. The UP then runs a virtual machine VM with (P, t, l, a), i.e. the VM runs P on a for t steps, and halts with an l bit output written on its output tape, and obtains y, the result of the computation. Finally, UP writes $y' = \langle y, P, t, K\rangle$ on the output tape and halts. The parameters P, t, K are output to allow the unhardening procedure $\mathcal{U}'$ to verify that the result of the computation is authentic. The output y' of UP is encoded, i.e. encrypted and/ or authenticated, by the OVM (the encoded value returned by the OVM is denoted ω). The function *createUP*, given a secret key K, generates and returns the Turing machine UP_K, represented as a string. The secret key K, is instantiated during the generation and is concatenated to the constant parts of the string. The parameter **K** is emphasised to separate it from other constant parameters of the string.

createUP(**K**) {

return "`read a'`

$(\mathtt{a}, \mathtt{t}, \mathtt{l}, \mathtt{c_P}) \leftarrow$ `parse a'`

$\mathtt{P} \leftarrow \mathtt{VD}(\text{“}\|[\mathbf{K}]\|\text{”}, \mathtt{c_P})$

$\mathtt{y} \leftarrow \mathtt{VM}(\mathtt{P}, \mathtt{t}, \mathtt{l}, \mathtt{a})$

`write y'` $= \langle \mathtt{y}, \mathtt{P}, \mathtt{t}, \text{“}\|[\mathbf{K}]\|\text{”}\rangle$”

}

Figure 15: The *createUP* function, creating the UP for a given K by string concatenation denoted by $\|$.

We show that given a *WBRPE* for $\{\mathrm{UP}_K\}$ we construct a *WBRPE* for any program securely.

Theorem 3.1 *Let* $\phi \in \{WB{-}IND{-}CPA, WB{-}UNF{-}OUT, (WB{-}UNF{-}PRG \,\&\, WB{-}IND{-}CPA)\}$ *and let* $\Pi = (\mathcal{G}_{AE}, \mathcal{AE}, \mathcal{VD})$ *is an* $IND-CPA$ *secure authenticated encryption scheme. If* $W = (\mathcal{G}, \mathcal{H}, \mathcal{U})$ *is a* ϕ *secure* WBRPE *scheme for the universal program UP, then* $W' = Univ(W)$ *is a* ϕ *secure* WBRPE *scheme for every program.*

Proof We prove the theorem for each of the three values of ϕ, in Lemmas 4.1, 4.11, 4.13 respectively. ∎

Theorem 3.2 *Let* $\phi \in \{WB-IND-CPA, WB-UNF-OUT, (WB-UNF-PRG \,\&\, WB-IND-CPA), WB-PRV\}$ *and let* $\Pi = (\mathcal{G}_{AE}, \mathcal{AE}, \mathcal{VD})$ *be an* $IND-CPA$ *secure authenticated encryption scheme. If* $W = (\mathcal{G}, \mathcal{H}, \mathcal{U})$ *is a* ϕ *secure* WBRPE *scheme with privacy of remote inputs for the universal program UP, then* $W' = Univ(W)$ *is a* ϕ *secure* WBRPE *scheme with privacy of remote inputs for every program.*

Proof We prove the theorem for $\phi = WB-PRV$ in Lemma 5.1, and present a proof sketch for $\phi = (WB-IND-CPA, WB-UNF-OUT, WB-UNF-PRG)$ in propositions 5.3, 5.4, 5.5.

Given a specific *WBRPE* scheme $W = (\mathcal{G}, \mathcal{H}, \mathcal{U})$ for UP we define the *Universal WBRPE* scheme $W' = (\mathcal{G}', \mathcal{H}', \mathcal{U}')$ for all programs, such that the scheme *W'* is defined as a function of $(\mathcal{G}, \mathcal{H}, \mathcal{U})$. Specifically, we present an efficient mapping from any *WBRPE* scheme for UP_d to a *Universal WBRPE* scheme. If W is a *WBRPE* scheme, then $W' = Univ(W)$ is a *Universal WBRPE* scheme such that $Univ(W) = (\mathcal{G}', \mathcal{H}', \mathcal{U}')$. See Figures 18, 19, 16, for detailed description of $Univ(W)$. Below is an informal description of the computations performed by *Univ*.

3.1 The Generation Procedure

The generation procedure $\mathcal{G}'$ of the *Universal WBRPE* scheme W' applies $\mathcal{G}$ of the specific *WBRPE* W and obtains the hk and vk, i.e. the hardening and the verification keys, and the OVM. It applies the $createOVM'$ function on the OVM of the specific *WBRPE* scheme to generate the OVM' of the *Universal WBRPE* scheme W' and returns the tuple $\langle hk, vk, \text{OVM'} \rangle$. See Figure 17. The $createOVM'$ function generates the OVM' Turing machine encoded in a string. The OVM' reads (c, a, t, l) of the input tape and generates an input for the OVM Turing machine. The OVM decodes c_{UP} and runs the universal program on input a', for t' steps and writes an l' bit output on its output tape, where t' comprised of the number of steps performed by UP, the number of steps the input program P is executed and of the number of steps it takes the virtual machine to execute P, i.e. bounded by some polynomial $p(\cdot)$ in t. The output length l' is the length of UP's output, which is the tuple $\langle y, P, t, K \rangle$.

The OVM' is comprised of two concatenated strings with the OVM.

3.2 The Hardening Procedure

The input to the hardening procedure $\mathcal{H}'$ of the *Universal WBRPE* scheme W' is a program P supplied by the local host. The universal hardening procedure first applies the generation procedure of the authenticated encryption scheme, e.g. in [14], obtains the secret key K and then encrypts the input program P using K, which results in c_P. Next, it generates the universal program, given the secret key K, and hardens it using $\mathcal{H}$ to obtain the pair c_{UP} and uk, subsequently returning the ordered pairs $\langle c_{UP}, c_P \rangle$ and $\langle uk, K \rangle$. Details in Figure 18.

We employ authenticated encryption in order to prevent forgery of the input programs, and to ensure that the input program P of the *Universal WBRPE* was not modified on transit, and replaced with some other input program P'.

3.3 The Unhardening Procedure

The unhardening procedure receives an ω and optional $[P, t]$ in an input. It applies the unhardening procedure $\mathcal{U}$ of the specific *WBRPE* scheme W on ω, and obtains the tuple $(y, \tilde{P}, \tilde{t}, \tilde{K})$. It then checks if the P, t parameters were supplied, if not it simply returns y, otherwise the validation of the input is also performed.

$\mathcal{G}'(1^k)$ {

$(hk, vk, \text{OVM}) \leftarrow \mathcal{G}(1^k)$

OVM' $\leftarrow createOVM'(\text{OVM})$

return $\langle hk, vk, \text{OVM'} \rangle$

}

Figure 16: Generation procedure of the *Universal WBRPE*

$createOVM'(OVM)$ {

return "`read (c,a,t,l)`

`(cUP, cP)` ← `parse c`

`a'` ← `(a, t, l, cP)`

`t'=p(t)+3`

`l'=l+|P|+|t|+|K|`

`write (cUP, a', t', l')`"

$\| OVM$

}

Figure 17: *createOVM'* function that generates the OVM'.

$\mathcal{U}'$ verifies that the pair (P, t) supplied by the adversarial algorithm and the pair $(\tilde{P}, \tilde{t})$ output from the universal program UP_K are identical, and that the secret key K from uk equals to the secret key $\tilde{K}$ from the output of UP. This is critical in order to verify that the result of the computation is authentic and not a forgery. If the result is authentic, the $\mathcal{U}$ is applied on the universal program UP_K t' and ω, such that UP_K and t' are generated from the input parameters supplied to $\mathcal{U}'$. These steps are performed in order to validate the result ω, i.e. that it is an authentic computation the universal program after a t' steps execution. The universal unhardening procedure returns y as its output. See the details of the implementation in Figure 19.

$\mathcal{H}'_{hk}(P)$ {

$K \leftarrow \mathcal{G}_E(1^k)$

$c_P \leftarrow \mathcal{E}_K(P)$

$\text{UP}_K \leftarrow createUP(K)$

$(c_{UP}, uk) \leftarrow \mathcal{H}_{hk}(\text{UP}_K)$

$c \leftarrow \langle c_{UP}, c_P \rangle$

$uk' \leftarrow \langle uk, K \rangle$

return (c, uk')

}

Figure 18: The hardening procedure of the *Universal WBRPE*.

$\mathcal{U}'_{uk',vk}(\omega, [P, t])$ {

$\langle uk, K \rangle \leftarrow uk'$

$\left\langle y, \tilde{P}, \tilde{t}, \tilde{K} \right\rangle \leftarrow \mathcal{U}_{uk,vk}(\omega)$

if $((P, t) = NULL)$ { return y }

if $((\tilde{P}, \tilde{t}, \tilde{K}) \neq (P, t, K))$ { return $\perp$ }

$\text{UP}_K \leftarrow createUP(K)$

$t' = t + 3$

$\left\langle y, \tilde{P}, \tilde{t}, \tilde{K} \right\rangle \leftarrow \mathcal{U}_{uk,vk}(\omega, UP_K, t')$

return y

}

Figure 19: The uhhardening procedure of the *Universal WBRPE*.

4 Analysis of the Universal WBRPE

In this section we present the construction of the *Universal WBRPE* scheme given a *WBRPE* scheme for a specific universal program UP presented in Section 3, Figure 15. We prove the security of the *Universal WBRPE* by reduction to the security of the underlying building block, the specific *WBRPE*. This is a first reduction between two white box primitives and it employs techniques of code manipulation, that are essential in white box security, since the programs are transfered for execution to remote host.

4.1 Indistinguishability

If the specific *WBRPE* provides indistinguishability for a specific program family UP_K, then the *Universal WBRPE* provides indistinguishability for any program.

Lemma 4.1 *(Indistinguishability) Let $W = (\mathcal{G}, \mathcal{H}, \mathcal{U})$ be an $WB - IND - CPA - \varphi$ secure WBRPE scheme for UP_d and let $\Pi = (\mathcal{G}_{AE}, \mathcal{AE}, \mathcal{VD})$ be an $IND - CPA$ secure encryption scheme. Then $W' = (\mathcal{G}', \mathcal{H}', \mathcal{U}')$ is an $WB - IND - CPA - \varphi$ secure WBRPE scheme for any program.*

Proof We reduce the security of the *Universal WBRPE* scheme W' to the security of the primitives that underlie the construction of W', i.e. the specific *WBRPE* W and the encryption scheme Π. More specifically, we show that given a PPT algorithm A' against W' we construct three PPT algorithms, AL, AR and A_Π, showing that the advantage that A' gains in the indistinguishability experiment $Exp^{WB-IND-CPA-\varphi}_{W',A'}(k)$ against W', is bounded by the sum of the advantages that AL and AR gain in indistinguishability experiment against W along with the advantage that A_Π gains in the indistinguishability experiment against Π. Therefore, if the advantages of these algorithms are negligible, their sum is also negligible, implying that the advantage of the algorithm A' against the *Universal WBRPE* W' is negligible. Details follow.

For every PPT algorithm A' let $\mathtt{Adv}^{WB-IND-CPA-\varphi}_{W',A'}(k)$ be the advantage that the algorithm A' gains during the $WB - IND - CPA - \varphi$ experiment against the *Universal WBRPE* W', as in definition 2.2:

$$\mathtt{Adv}^{WB-IND-CPA-\varphi}_{W',A'}(k) = 2 * \Pr[\mathtt{Exp}^{WB-IND-CPA-\varphi}_{W',A'}(k) = 1] - 1$$

Where the probabilities are taken over $\mathcal{G}'$ and A''s coins tosses.
In the $\mathtt{Exp}^{WB-IND-CPA-\varphi}_{W',A'}(k) = 1$ experiment, the algorithm A' picks two challenge programs P_0 and P_1, then the experiment tosses a bit, and the corresponding program is hardened. Then the challenge $(c_{UP}, \mathcal{AE}_e(P_b))$ is given to A', which has to distinguish which of the input programs was hardened, i.e. if it is a hardening of P_0 or of P_1.

Given a PPT algorithm $A' = (A'_1, A'_2)$ we construct PPT algorithms $AR = (AR_1, AR_2)$ and $AL = (AL_1, AL_2)$ against the specific *WBRPE* scheme W, and a PPT algorithm $A_\Pi = (A_{1,\Pi}, A_{2,\Pi})$ against Π. The proof follows from the following inequality

$$\mathtt{Adv}^{WB-IND-CPA-\varphi}_{W',A'}(k) \leq$$
$$\leq \mathtt{Adv}^{WB-IND-CPA-\varphi}_{W,AR}(k) + \mathtt{Adv}^{WB-IND-CPA-\varphi}_{\Pi,A_\Pi}(k) + \mathtt{Adv}^{WB-IND-CPA-\varphi}_{W,AL}(k) \quad (1)$$

The proof that inequality 1 holds, follows from claim 4.2. ∎

Claim 4.2 *The advantage $\mathtt{Adv}^{WB-IND-CPA-\varphi}_{W',A'}(k)$ of the PPT algorithm $A = (A_1, A_2)$ is bounded by the sum of the advantages of the algorithms AR, AL and A_Π against the schemes W and Π respectively:*

$$\mathtt{Adv}^{WB-IND-CPA-\varphi}_{W',A'}(k) \leq \mathtt{Adv}^{WB-IND-CPA-\varphi}_{W,AL}(k) + \mathtt{Adv}^{IND-CPA-\varphi}_{\Pi,A_\Pi}(k) + \mathtt{Adv}^{WB-IND-CPA-\varphi}_{W,AR}(k)$$

Proof From proposition 4.3 below follows that:

$$\mathtt{Adv}^{WB-IND-CPA-\varphi}_{W',A'}(k) = \Big|\Pr[Exp^{WB-IND-CPA-\varphi}_{W',A'}(k)=0|b=0] - \Pr[Exp^{WB-IND-CPA-\varphi}_{W',A'}(k)=1|b=1]\Big|$$

By triangle inequality we obtain:

$$\begin{aligned}
&\Big|\Pr[Exp^{WB-IND-CPA-\varphi}_{W',A'}(k)=0|b=0] - \Pr[Exp^{WB-IND-CPA-\varphi}_{W',A'}(k)=1|b=1]\Big| \leq \\
&\leq \Big|\Pr[Exp^{WB-IND-CPA-\varphi}_{W',A'}(k)=0|b=0] - \Pr[Exp^{WB-IND-CPA-\varphi}_{W,AL}(k)=1|b=1]\Big| + \\
&+\Big|\Pr[Exp^{WB-IND-CPA-\varphi}_{W,AL}(k)=1|b=1] - \Pr[Exp^{IND-CPA-\varphi}_{\Pi,A_\Pi}(k)=1|b=1]\Big| + \\
&+\Big|\Pr[Exp^{IND-CPA-\varphi}_{\Pi,A_\Pi}(k)=1|b=1] - \Pr[Exp^{WB-IND-CPA-\varphi}_{W',A'}(k)=1|b=1]\Big|
\end{aligned}$$

We proof follows from claims 4.4, 4.6, 4.8. ∎

Proposition 4.3

$$\mathit{Adv}^{WB-IND-CPA-\varphi}_{W',A'}(k) = \Big|\Pr[\mathit{Exp}^{WB-IND-CPA-\varphi}_{W',A'}(k)=0|b=0] - \Pr[\mathit{Exp}^{WB-IND-CPA-\varphi}_{W',A'}(k)=1|b=1]\Big|$$

Proof By Definition 2.2, $\mathtt{Adv}^{WB-IND-CPA-\varphi}_{W',A'}(k) = 2 * \Pr[\mathtt{Exp}^{WB-IND-CPA-\varphi}_{W',A'}(k) = 1] - 1$. Denote for simplicity $E \equiv \mathtt{Exp}^{WB-IND-CPA-\varphi}_{W',A'}(k)$. Then

$$\begin{aligned}
&2 * \Pr[E=1] - 1 = \Pr[E=1] + \Pr[E=1] - 1 = \\
&\quad = \Pr[E=1] - (1 - \Pr[E=1]) = \Pr[E=1] - \Pr[E=0] = \\
&\quad = \Pr[E=1|b=0] * \Pr[b=0] + \Pr[E=1|b=1] * \Pr[b=1] - \\
&\qquad\qquad -(\Pr[E=0|b=0] * \Pr[b=0] + \Pr[E=0|b=1] * \Pr[b=1]) = \\
&\quad = \Pr[E=1|b=0] * \Pr[b=0] + \Pr[E=1|b=1] * \Pr[b=1] - \\
&\qquad\qquad - \Pr[E=0|b=0] * \Pr[b=0] - \Pr[E=0|b=1] * \Pr[b=1] = \\
&\quad = \frac{1}{2}(\Pr[E=1|b=1] - \Pr[E=0|b=0] + \Pr[E=1|b=0] - \Pr[E=0|b=1]) = \\
&\quad = \frac{1}{2}(\Pr[E=1|b=1] - \Pr[E=0|b=0] + 1 - \Pr[E=0|b=0] - 1 + \Pr[E=1|b=1]) = \\
&\quad = \frac{1}{2}(2 * \Pr[E=1|b=1] - 2 * \Pr[E=0|b=0]) = \Pr[E=1|b=1] - \Pr[E=0|b=0]
\end{aligned}$$

∎

Claim 4.4 *Let $W' = (\mathcal{G}', \mathcal{H}', \mathcal{U}')$ be a* Universal WBRPE *scheme. For any pair of PPT algorithms $A' = (A'_1, A'_2)$, there exists a pair of PPT algorithm $AL = (AL_1, AL_2)$ s.t. holds:*

$$\mathit{Adv}^{WB-IND-CPA-\varphi}_{W,AL}(k) \geq \Big|\Pr[Exp^{WB-IND-CPA-\varphi}_{W',A'}(k)=0|b=0] - \Pr[Exp^{WB-IND-CPA-\varphi}_{W,AL}(k)=1|b=1]\Big|$$

Proof Given a PPT algorithm $A' = (A'_1, A'_2)$ against the *Universal WBRPE* scheme W' we construct a PPT algorithm $AL = (AL_1, AL_2)$ against the $WB - IND - CPA - \varphi$ secure specific *WBRPE* scheme W, s.t.

$$\begin{aligned}
&\Big|\Pr[Exp^{WB-IND-CPA-\varphi}_{W,AL}(k)=0|b=0] - \Pr[Exp^{WB-IND-CPA-\varphi}_{W,AL}(k)=1|b=1]\Big| \geq \\
&\geq \Big|\Pr[Exp^{WB-IND-CPA-\varphi}_{W',A'}(k)=0|b=0] - \Pr[Exp^{WB-IND-CPA-\varphi}_{W,AL}(k)=1|b=1]\Big|
\end{aligned}$$

The algorithm AL is presented in Figures 4.1, 4.1. According to proposition 4.3 the advantage of AL is:

$$\mathtt{Adv}_{W,AL}^{WB-IND-CPA-\varphi}(k) = \left| \Pr[Exp_{W,AL}^{WB-IND-CPA-\varphi}(k) = 0 | b = 0] - \Pr[Exp_{W,AL}^{WB-IND-CPA-\varphi}(k) = 1 | b = 1] \right|$$

The algorithm AL obtains an oracle access to hardening procedure $\mathcal{HO}$ of the specific *WBRPE* scheme W and black box access to A'.

We reduce the security of the *Universal WBRPE* to the security of the specific *WBRPE* when the bit that the experiment of the specific *WBRPE* tosses is always 0, i.e. $b = 0$.

During the simulation, A' obtains black box access to $\mathcal{HP}'$ as defined in Figure 21. The $\mathcal{HP}'$ is a hardening procedure of the *Universal WBRPE* scheme W' that AL simulates for A', using the hardening oracle $\mathcal{HO}$ of the specific *WBRPE* scheme W and the encryption scheme Π, which it has in its possession.

In proposition 4.5 we show that the simulation run by AL is identical to the execution of the experiment of A', when the challenge bit b is 0. Namely, when $b = 0$, the view of A' when invoked by our implementation of the algorithm AL, is identical to its view in its original experiment, when $b = 0$ and the challenge string that it obtains is always of the form $(c_{UP}, \mathcal{E}_e(P_0))$ where $c_{UP} \leftarrow \mathcal{H}_{hk}(\mathrm{UP}_K).c$ and K is a key generated by $\mathcal{G}_{AE}$.

Since the view of the algorithm A', when invoked by AL against W, is identical to its view in the indistinguishability experiment against W', it performs the same computation and therefore returns the same output. The claim follows.

Algorithm $AL_1^{\mathcal{HO}(\cdot)}(1^k, \mathrm{OVM}, vk, \varphi)\{$

$K \xleftarrow{R} \mathcal{G}_{AE}(1^k)$
$K' \in_R \{0,1\}^k$
$\mathrm{UP}_K \leftarrow createUP(K)$
$\mathrm{UP}_{K'} \leftarrow createUP(K')$
$s \leftarrow \langle 1^k, \mathrm{OVM}, vk, K, K', \varphi \rangle$
return $(\mathrm{UP}_K, \mathrm{UP}_{K'}, s)$

$\}$

Algorithm $AL_2^{\mathcal{HO}(\cdot)}(c_{UP}, s)\{$

$\langle 1^k, \mathrm{OVM}, vk, K, K', \varphi \rangle \leftarrow parse\ s$
OVM' $\leftarrow createOVM'(\mathrm{OVM})$
$(P_0, P_1, s') \leftarrow {A'}_1^{\mathcal{HP}'(\cdot)}(1^k, \mathrm{OVM'}, vk)$
$c_P \leftarrow \mathcal{AE}_K(P_0)$
$c \leftarrow \langle c_{UP}, c_P \rangle$
return ${A'}_2^{\mathcal{HP}'(\cdot)}(c, s')$

$\}$

Figure 20: Implementation of $AL = (AL_1, AL_2)$ using $A' = (A'_1, A'_2)$ in $Exp_{W,AL}^{WB-IND-CPA-\varphi}(k)$.

Proposition 4.5 *Let A',AL be as in claim 4.4. Then:*

$$\Pr[Exp_{W,AL}^{WB-IND-CPA-\varphi}(k) = 0 | b = 0] \geq \Pr[Exp_{W',A'}^{WB-IND-CPA-\varphi}(k) = 0 | b = 0]$$

Proof When AL is invoked in the indistinguishability experiment of the *Universal WBRPE* scheme W', it runs A', simulating for it the indistinguishability experiment of the *Universal WBRPE* scheme W', eventually returning A' output.

We do not make any assumption on the internal behaviour of the algorithm A' but only outline the following observation, given a sequence of random coins the execution of the algorithm A' is a deterministic function of the inputs. Therefore the computation performed by A' is determined by a random string, which the experiment supplies to it, and by the responses that it obtains to its hardening oracle queries. Hence if AL accurately simulates the steps of the indistinguishability experiment of W' for A', and implements the hardening procedure for A' as specified in the construction of the $\mathcal{H}'$, then given the same random string r, A' will see the same execution in both cases, and will therefore perform the same computation and

Hardening $\mathcal{HP}'^{\mathcal{HO}(\cdot,hk,\varphi)}(P,\varphi)$

$K \xleftarrow{R} \mathcal{G}_{AE}(1^k)$
$c_P \leftarrow \mathcal{AE}_K(P)$
$\mathrm{UP}_K \leftarrow createUP(K)$
if $(\varphi = PK)$
    return $\mathcal{HO}(\mathrm{UP}_K, hk, \varphi)$
else
    $(c_{UP}, uk) \leftarrow \mathcal{HO}(\mathrm{UP}_K, hk, \varphi)$
    $c \leftarrow \langle c_{UP}, c_P \rangle$
    return $(c, \langle uk, K \rangle)$

Figure 21: The hardening procedure $\mathcal{HP}'$ of algorithm AL, using oracle $\mathcal{HO}$.

subsequently return the same results. Hence when the challenge bit is $b = 0$ the view of A' when invoked by A is distributed identically to its view in $Exp^{WB-IND-CPA-\varphi}_{W',A'}(k)$ and in both executions the hardened challenge that it obtains is of the form: $(c_{UP}, \mathcal{E}_e(P_0))$. Therefore in both cases A' will return the same response and the proposition follows. ∎

Claim 4.6 *Let* $W' = (\mathcal{G}', \mathcal{H}', \mathcal{U}')$ *be a* Universal WBRPE *scheme. For any pair of PPT algorithms* $A' = (A'_1, A'_2)$ *there exists a pair of PPT algorithms* $A_\Pi = (A_{1,\Pi}, A_{2,\Pi})$ *s.t. holds:*

$$\mathtt{Adv}^{IND-CPA-\varphi}_{\Pi,A_\Pi}(k) \geq \left| \Pr[Exp^{WB-IND-CPA-\varphi}_{W,AL}(k) = 0 | b = 0] - \Pr[Exp^{IND-CPA-\varphi}_{\Pi,A_\Pi}(k) = 1 | b = 1] \right|$$

Proof Given a PPT algorithm $A' = (A'_1, A'_2)$ that gains advantage $\mathtt{Adv}^{WB-IND-CPA-\varphi}_{W',A'}(k)$ in the indstinguishability experiment of the *Universal WBRPE* W', construct a PPT algorithm $A_\Pi = (A_{1,\Pi}, A_{2,\Pi})$, that gains a related advantage $\mathtt{Adv}^{IND-CPA-\varphi}_{\Pi,A_\Pi}(k)$ in the indistinguishability experiment of the authenticated encryption scheme Π as defined in [14], s.t.

$$\left| \Pr[Exp^{IND-CPA-\varphi}_{\Pi,A_\Pi}(k) = 0 | b = 0] - \Pr[Exp^{IND-CPA-\varphi}_{\Pi,A_\Pi}(k) = 1 | b = 1] \right| \geq$$

$$\geq \left| \Pr[Exp^{WB-IND-CPA-\varphi}_{W,AL}(k) = 1 | b = 1] - \Pr[Exp^{IND-CPA-\varphi}_{\Pi,A_\Pi}(k) = 1 | b = 1] \right|$$

The algorithm A_Π is presented in Figures 4.1, 4.1. According to definition 2.2 and proposition 4.3, the advantage of A_Π is:

$$\mathtt{Adv}^{IND-CPA-\varphi}_{\Pi,A_\Pi}(k) = \left| \Pr[Exp^{IND-CPA-\varphi}_{\Pi,A_\Pi}(k) = 0 | b = 0] - \Pr[Exp^{IND-CPA-\varphi}_{\Pi,A_\Pi}(k) = 1 | b = 1] \right|$$

The algorithm A_Π obtains an oracle access to the encryption $\mathcal{AEO}$ oracle of the authenticated encryption scheme Π and black box access to A'. We reduce the security of the *Universal WBRPE* W' to the security of the authenticated encryption scheme Π when the challenge bit b that is chosen by the experiment of W' is 0. In this case the challenge message that the algorithm A' obtains is of the form $(c_{UP'}, \mathcal{AE}_K(P_0))$, where $c_{UP'} \leftarrow \mathcal{H}_{hk}(\mathrm{UP}_{K'}).c$ and $K' \in \{0,1\}^k$ is not the secret key that was used to encrypt the challenge. During the simulation A' obtains a black box access to the $\mathcal{HP}'$ defined in Figure 23, that A_Π simulates for it, using the authenticated encryption scheme Π and the specific *WBRPE* scheme W which it has in

its possession. In proposition 4.7 we show that when the challenge bit $b = 0$ in the indistinguishability experiment for Π, the simulation run by A_Π is identical to the simulation that AL runs for A' in the indistinguishability experiment against *WBRPE* scheme W, when the challenge bit is $b = 1$. Therefore, the view of A' is distributed identically to its view when invoked by AL and the ciphertext that it sees is of the form $(c_{UP'}, \mathcal{E}_K(P_0))$ where $c_{UP'}$ is the result of the hardening procedure $\mathcal{H}$ applied on $\mathrm{UP}_{K'}$ and $K' \in_R \{0,1\}^k$.

Since the view of the algorithm A', when invoked by A_Π against Π, is identical to its view when invoked by AL against W, it performs the same computation and therefore returns the same output. The claim follows. ∎

Algorithm $A_{1,\Pi}^{\mathcal{AEO}(\cdot)}(1^k)$

$(hk, vk, \mathrm{OVM}) \leftarrow \mathcal{G}(1^k)$
$\mathrm{OVM'} \leftarrow createOVM'(\mathrm{OVM})$
$(P_0, P_1, s') \leftarrow A'^{\mathcal{HP}'(\cdot)}_1(1^k, \mathrm{OVM'}, vk)$
$s \leftarrow \langle hk, vk, \mathrm{OVM'}\rangle$
return $(P_0, P_1, \langle s, s'\rangle)$

Algorithm $A_{2,\Pi}^{\mathcal{AEO}(\cdot)}(c_P, \langle s, s'\rangle)$

$\langle hk, vk, \mathrm{OVM'}\rangle \leftarrow parse\ s$
$K' \in_R \{0,1\}^k$
$\mathrm{UP}_{K'} \leftarrow createUP(K')$
$(c_{UP'}, uk) \leftarrow \mathcal{H}_{hk}(\mathrm{UP}_{K'})$
$c \leftarrow \langle c_{UP'}, c_P\rangle$
return $A'^{\mathcal{HP}'(\cdot)}_2(c, s')$

Figure 22: The implementation of $A = (A_{1,\Pi}, A_{2,\Pi})$ using $A' = (A'_1, A'_2)$ in $Exp^{IND-CPA}_{\Pi,A_\Pi}(k)$.

Proposition 4.7 *Let AL, A_Π be as in claim 4.6. Then:*

$$\Pr[Exp^{IND-CPA-\varphi}_{\Pi,A_\Pi}(k) = 0 | b = 0] \geq \Pr[Exp^{WB-IND-CPA-\varphi}_{W,AL}(k) = 1 | b = 1]$$

Proof Given a sequence of random coins r the simulation is determined, and maps the computation for the specific r to the same result. The computations performed by A' are determined by the sequence of random coins and the inputs that it obtains in response for its queries. Hence, if A_Π accurately simulates the steps specified in the indistinguishability experiment of W' for A', and implements the hardening procedure for A' as specified in the construction of $\mathcal{H}'$, then given the same random string r, A' will see the same execution in both cases and will therefore perform the same computation and subsequently return the same results. Therefore, when the challenge bit b is 0 in the indistinguishability experiment of A_Π against Π, the view of A' is identical to its view in the simulation that AL runs for A' in the indistinguishability experiment against the *WBRPE* scheme W and the challenge bit b is 1 and in both executions the hardened challenge that it obtains is of the form $(c_{UP'}, \mathcal{E}_K(P_0))$. Hence, holds:s

$$\mathtt{Adv}^{IND-CPA-\varphi}_{\Pi,A_\Pi}(k) \geq \left| \Pr[Exp^{WB-IND-CPA-\varphi}_{W,AL}(k) = 1 | b = 1] - \Pr[Exp^{IND-CPA-\varphi}_{\Pi,A_\Pi}(k) = 1 | b = 1] \right|$$

The proof is similar to the proof of proposition 4.5. ∎

Claim 4.8 *Let $W' = (\mathcal{G}', \mathcal{H}', \mathcal{U}')$ be a* Universal WBRPE *scheme. For any pair of PPT algorithms $A' = (A'_1, A'_2)$ there exists a pair of PPT algorithms $AR = (AR_1, AR_2)$, s.t. holds:*

$$\mathtt{Adv}^{WB-IND-CPA-\varphi}_{W,AR}(k) \geq \left| \Pr[Exp^{IND-CPA-\varphi}_{\Pi,A_\Pi}(k) = 1 | b = 1] - \Pr[Exp^{WB-IND-CPA-\varphi}_{W',A'}(k) = 1 | b = 1] \right|$$

Hardening Procedure $\mathcal{HP}'(P, hk, \varphi)$

$K \leftarrow \mathcal{G}_{AE}(1^k)$
if $(\varphi = PK)$
    return hk
else
    $c_P \leftarrow \mathcal{AE}_K(P)$
    $\mathrm{UP}_K \leftarrow createUP(K)$
    $(c_{UP}, uk) \leftarrow \mathcal{H}_{hk}(\mathrm{UP}_K)$
    $c \leftarrow \langle c_{UP}, c_P \rangle$
    return $(c, \langle uk, K \rangle)$

Figure 23: The implementation of the hardening procedure $\mathcal{HP}'$.

Proof Given a PPT algorithm $A' = (A'_1, A'_2)$ against the *Universal WBRPE* scheme W' we construct a PPT algorithm $A = (A_1, A_2)$ against the $WB - IND - CPA - \varphi$ secure specific *WBRPE* W, s.t.

$$\left| \Pr[Exp^{WB-IND-CPA-\varphi}_{W,AR}(k) = 0 | b = 0] - \Pr[Exp^{WB-IND-CPA-\varphi}_{W,AR}(k) = 1 | b = 1] \right| \geq$$

$$\geq \left| \Pr[Exp^{IND-CPA-\varphi}_{\Pi,A_\Pi}(k) = 1 | b = 1] - \Pr[Exp^{WB-IND-CPA-\varphi}_{W',A'}(k) = 1 | b = 1] \right|$$

The algorithm AR is presented in Figures 4.1, 4.1. According to proposition 4.3, the advantage of AR is:

$$\mathtt{Adv}^{WB-IND-CPA-\varphi}_{W,AR}(k) = \left| \Pr[Exp^{WB-IND-CPA-\varphi}_{W,AR}(k) = 0 | b = 0] - \Pr[Exp^{WB-IND-CPA-\varphi}_{W,AR}(k) = 1 | b = 1] \right|$$

The algorithm AR obtains an oracle access to hardening procedure $\mathcal{HO}$ of the specific *WBRPE* scheme W and a black box access to A'. In this step we reduce the security of the *Universal WBRPE* scheme W' to the security of the specific *WBRPE* scheme W, where the bit that the experiment of the specific *WBRPE* chooses is always 1, i.e. $b = 1$.

During the simulation A' obtains black box access to $\mathcal{HP}'$ defined in Figure 21, which is the hardening procedure of the *Universal WBRPE* scheme W', that AR simulates for it, using the hardening oracle $\mathcal{HO}$ of the specific *WBRPE* scheme W and the encryption scheme Π, which it has in its possession. In proposition 4.9, we claim that the simulation run by AR is identical to the execution of the experiment of A', when the challenge bit in $\Pr[Exp^{WB-IND-CPA-\varphi}_{W,AR}(k)$ is $b = 0$, i.e. the view of A' when invoked by our implementation of the algorithm AR is identical to its view in its original experiment $\Pr[Exp^{WB-IND-CPA-\varphi}_{W',A'}(k)$ when $b = 1$, and the challenge string that A' obtains is of the form $(c_{UP}, \mathcal{E}_K(P_1))$, where $c_{UP} \leftarrow \mathcal{H}_{hk}(\mathrm{UP}_K).c$ and K is a key generated by $\mathcal{G}_{AE}$. Furthermore, in proposition 4.10, we claim that when $b = 1$, the view of A' when invoked by the simulation run by AR, is identical to its view in the simulation of A_Π, and the ciphertext that it obtains is of the form $(c_{UP'}, \mathcal{E}_K(P_1))$, where $c_{UP'} \leftarrow \mathcal{E}_K(\mathrm{UP}_{K'})$ and $K' \in_R \{0,1\}^k$.

Since the view of the algorithm A' when invoked by AR against W, is identical to its view in the indistinguishability experiment against W' or to its view in the simulation of A_Π, it performs the same computation and therefore returns the same output. The claim follows. ∎

Proposition 4.9 *Let A', AR be as in claim 4.8, then:*

$$\Pr[Exp^{WB-IND-CPA-\varphi}_{W,AR}(k) = 0 | b = 0] \geq \Pr[Exp^{WB-IND-CPA-\varphi}_{W',A'}(k) = 1 | b = 1]$$

Algorithm $AR_1^{\mathcal{HO}(\cdot)}(1^k, \text{OVM}, vk, \varphi)$

$K \leftarrow \mathcal{G}_{AE}(1^k)$
$K' \in_R \{0,1\}^k$
$\text{UP}_K \leftarrow createUP(K)$
$\text{UP}_{K'} \leftarrow createUP(K')$
$s \leftarrow \langle 1^k, \text{OVM}, vk, K, K' \rangle$
return $(\text{UP}_K, \text{UP}_{K'}, s)$

Algorithm $AR_2^{\mathcal{HO}(\cdot)}(c_{UP}, s)$

$\langle 1^k, \text{OVM}, vk, K, K' \rangle \leftarrow parse\ s$
OVM' $\leftarrow createOVM'(\text{OVM})$
$(P_0, P_1, s') \leftarrow A_1'^{\mathcal{HP}'(\cdot)}(1^k, \text{OVM}, vk)$
$c_P \leftarrow \mathcal{AE}_K(P_1)$
$c \leftarrow \langle c_{UP}, c_P \rangle$
return $A_2'^{\mathcal{HP}'(\cdot)}(c, s')$

Figure 24: Implementation of $AR = (AR_1, AR_2)$ using $A' = (A_1', A_2')$ in $Exp_{W,AR}^{WB-IND-CPA-\varphi}(k)$

Proposition 4.10 *Let A_Π, AR be as in claim 4.8, then:*

$$\Pr[Exp_{W,AR}^{WB-IND-CPA-\varphi}(k) = 1 | b = 1] \geq \Pr[Exp_{\Pi,A\Pi}^{WB-IND-CPA-\varphi}(k) = 1 | b = 1]$$

4.2 Unforgeability Specification

If the *WBRPE* provides output unforgeability for a specific program family UP_K, then the *Universal WBRPE* provides output unforgeability for any program. In contrast to achieve program unfogreability we need the specific *WBRPE* to provide indistinguishability of inputs. Intuitively, if the adversary can observe the secret key K inside the universal program, then it can forge a tuple $(c_{UP}, \mathcal{E}_K(P))$ for some P'.

Lemma 4.11 *(Correct Output) Let $W = (\mathcal{G}, \mathcal{H}, \mathcal{U})$ be an $WB - UNF - OUT - \varphi$ secure* WBRPE *scheme for UP and let $\Pi = (\mathcal{G}_{AE}, \mathcal{AE}, \mathcal{VD})$ be a secure authenticated encryption scheme. Then $W' = (\mathcal{G}', \mathcal{H}', \mathcal{U}')$ is a $WB - UNF - OUT - \varphi$ secure* WBRPE *scheme for any program.*

Proof The proof of lemma follows from claim 4.12. ∎

Claim 4.12 *For every PPT algorithm A' and any program P, let $\mathtt{Adv}_{W',A'}^{WB-UNF-OUT-\varphi}(k)$ be the advantage that the algorithm A' gains during the $WB - UNF - OUT - \varphi$ experiment against W', then*

$$\mathtt{Adv}_{W,A}^{WB-UNF-OUT-\varphi}(k) \geq \mathtt{Adv}_{W',A'}^{WB-UNF-OUT-\varphi}(k)$$

Proof Given a PPT algorithm $A' = (A_1', A_2')$ against the *Universal WBRPE* scheme W', we construct a PPT algorithm $A = (A_1, A_2)$ against the *WBRPE* scheme W in the $\mathbf{Exp}_{W,A}^{WB-UNF-OUT-\varphi}(k)$ experiment. The algorithm A obtains an oracle access to A' and to the hardening functionality $\mathcal{HO}$ of the *WBRPE* scheme and simulates the execution of the *Universal WBRPE* $UNF - OUT - \varphi$ experiment for A'. The implementation of A and $\mathcal{HP}'$ in 25, 26.

Next, we show that the view of A' when run by A is distributed identically to its view when invoked in the environment of its experiment Furthermore, given the successful forgery of the *Universal WBRPE* scheme, which is output by A', A can essentially construct a forgery for WBRPE scheme. This holds only if A precisely simulates the experiment for A' and if the result output by A complies with the definition of output forgery, defined in 4.

Given a random coins sequence $\mathbf{r}$ the execution is determined and therefore the experiment computes a deterministic function of the input. Given the same random coins, the experiment of *WBRPE* for A maps the inputs to the same output values, and thus performs the same computation.

Upon initiation of the WBRRPE $WB - UNF - OUT - \varphi$ experiment, first the keys hk, vk and the OVM are generated and A is invoked with the security parameter, the constant string OVM, and the public

Algorithm $A^{\mathcal{HO}(\cdot,hk,\varphi)}(1^k, \text{OVM}, vk)\{$

OVM' $\leftarrow createOVM'(\text{OVM})$

$(\omega, \widehat{P}, \widehat{t}, \widehat{uk}) \leftarrow A'^{\mathcal{HP}'(\cdot,hk,\varphi)}(1^k, \text{OVM'}, vk)$

$\left\langle y, \tilde{P}, \tilde{t}, \tilde{K} \right\rangle \leftarrow \mathcal{U}_{\widehat{uk},vk}(\omega)$

$\text{UP}_{\tilde{K}} \leftarrow createUP(\tilde{K})$

$t' = \widehat{t} + 3$

return $(\omega, \text{UP}_{\tilde{K}}, t', \widehat{uk})$

$\}$

Figure 25: The implementation of A in $Exp^{WB-UNF-OUT}_{W,A}(k)$.

Hardening Procedure $\mathcal{HP}'(P, hk, \varphi)\{$

$K \xleftarrow{R} \mathcal{G}_{AE}(1^k)$

$c_P \leftarrow \mathcal{AE}_K(P)$

$\text{UP}_K \leftarrow createUP(K)$

$(c_{UP}, uk) \leftarrow \mathcal{HO}(\text{UP}_K)$

$c \leftarrow \langle c_{UP}, c_P \rangle$

return $(c, \langle uk, K \rangle)$

$\}$

Figure 26: The implementation of $\mathcal{HP}'$.

verification key vk. As its first step A generates the OVM' for A' and runs A' with the newly generated OVM' and the rest of the parameters which it obtained in an input.
For each hardening query that A' submits, A simulates for A' the hardening oracle of the *Universal WBRPE* scheme. $\mathcal{HP}'$ generates the encryption key pair and encrypts the input program P. Then UP_K is generated and the hardening oracle is queried with this universal program. Then A' obtains a pair (c, uk). Since the same computation is performed by the hardening procedure $\mathcal{H}'$ of the *Universal WBRPE*, when both executions are given the same random string, the same parameters are generated in both executions and both the $\mathcal{HP}'$ and the $\mathcal{HO}'$ perform the same computation and therefore in both executions A' will see the same values. Clearly the view of A' is distributed identically in both experiments and therefore A' will perform the same computation in both cases. Which implies that it generates a successful forgery with the same probability in both executions.

Given the forgery of the *Universal WBRPE* scheme W' output by A', the result computed by A is a forgery of *WBRPE* scheme W. When A receives a tuple (ω, P, t, uk) from A' it computes the unhardening using the unhardening key uk and obtains $\left\langle y, \tilde{P}, \tilde{t}, K \right\rangle$ which is the output of the universal program. Next, given the secret key K it constructs the universal program (it can do this, since the design of the universal program is public), computes the number of execution steps t' for UP_K with t and returns $(\omega, \text{UP}_K, t', uk)$.

To check if the tuple given by A constitutes a successful forgery, the WBRPE $WB-UNF-OUT-\varphi$ experiment first unhardens and validates the forgery $\left\langle y, \tilde{P}, \tilde{t}, K \right\rangle \leftarrow \mathcal{U}_{uk,vk}(\omega, \text{UP}_K, t')$ consequently performing the following test

$$\forall a', \ \left\langle y, \tilde{P}, \tilde{t}, K \right\rangle \neq \text{UP}_{K_{t', |y, \tilde{P}, \tilde{t}, K|}}(a')$$

Which of course holds except with a negligible probability, since the input program P constitutes part of the output. ∎

Lemma 4.13 *(Correct Program) Let $W = (\mathcal{G}, \mathcal{H}, \mathcal{U})$ be an $WB-IND-CPA-SK$ and an $WB-UNF-PRG$ secure WBRPE scheme for UP and let $\Pi = (\mathcal{G}_{AE}, \mathcal{AE}, \mathcal{VD})$ be an $WB-IND-CPA$ authenticated encryption scheme. Then $W' = (\mathcal{G}', \mathcal{H}', \mathcal{U}')$ is a $WB-UNF-PRG$ secure WBRPE scheme for any program.*

Proof The proof of lemma follows from claim 4.14. ∎

Claim 4.14 *For every PPT algorithm A' and any program P, let $\mathbf{Adv}^{WB-UNF-PRG}_{W',A'}(k)$ be the advantage that the algorithm A' gains during the $WB-UNF-PRG$ experiment against W', then*

$$\mathbf{Adv}^{WB-UNF-PRG}_{W,A}(k) \geq \mathbf{Adv}^{WB-UNF-PRG}_{W',A'}(k)$$

Proof Given a PPT algorithm $A' = (A'_1, A'_2)$ against the *Universal WBRPE* scheme W', construct a PPT algorithm $A = (A_1, A_2)$ against the *WBRPE* scheme W in the $\mathbf{Exp}^{WB-UNF-PRG}_{W,A}(k)$ experiment. A obtains oracle access to A' and to the hardening procedure $\mathcal{HO}$ of the *WBRPE* scheme and simulates the execution of the *Universal WBRPE* $WB-UNF-PRG$ experiment for A'. The implementation of the algorithm A and $\mathcal{HP}'$ is in Figures 27, 28. We next show that the view of A' when run by A is distributed identically

Algorithm $A^{\mathcal{HO}(\cdot)}(1^k, \text{OVM}, vk)\{$

OVM' $\leftarrow createOVM'(\text{OVM})$

$(\omega, \widehat{P}, \widehat{t}, \widehat{uk}) \leftarrow A'^{\mathcal{HP}'(\cdot)}(1^k, \text{OVM'}, vk)$

$\left\langle y, \tilde{P}, \tilde{t}, \tilde{K} \right\rangle \leftarrow \mathcal{U}_{\widehat{uk},vk}(\omega)$

$\text{UP}_{\tilde{K}} \leftarrow createUP(\tilde{K})$

$t' = \widehat{t} + 3$

return $(\omega, \text{UP}_{\tilde{K}}, t', \widehat{uk})$

$\}$

Figure 27: The implementation of A in $Exp^{WB-UNF-PRG}_{W,A}(k)$.

Hardening Procedure $\mathcal{HP}'(hk, P, \varphi)\{$

$K \xleftarrow{R} \mathcal{G}_{AE}(1^k)$

$c_P \leftarrow \mathcal{AE}_K(P)$

$\text{UP}_K \leftarrow createUP(K)$

$(c_{UP}, uk) \leftarrow \mathcal{HO}(\text{UP}_K)$

$c \leftarrow \langle c_{UP}, c_P \rangle$

return $(c, \langle uk, K \rangle)$

$\}$

Figure 28: The implementation of $\mathcal{HP}'$.

to its view when invoked in the environment of its experiment. Furthermore, given a successful forgery of the *Universal WBRPE* scheme, which is output by A', A can construct a forgery for *WBRPE* scheme. This holds only if A precisely simulates the experiment for A' and if the result output by A complies with the definition of output forgery defined in 5.

Given the random coins sequence $\mathbf{r}$ the execution is determined and the experiment computes a deterministic function of the input. Furthermore, we do not make any assumption on the internal behaviour of the adversary but only claim that given a random string its computation is a deterministic function of the inputs that it sees. Therefore, given a random $\mathbf{r}$, when the $\mathcal{HP}'$ is implemented according to the construction and A accurately follows the steps of the experiment defined in 5, the advantage of A is identical to the advantage of A'. This is because the view of A' is distributed identically in both experiments and therefore A' will perform the same computation in both cases. Moreover, if A' generates a successful forgery of the *Universal WBRPE*, A outputs a successful forgery of the specific *WBRPE*. When A receives a tuple (ω, P, t, uk) from A' it computes the unhardening using the unhardening key uk and obtains $\left\langle y, \tilde{P}, \tilde{t}, K \right\rangle$ which is the output of the universal program. Next, given the secret key K it constructs the universal program (it can do this, since the design of the universal program is public), computes the number of execution steps t' for UP_K with t and returns $(\omega, \text{UP}_d, t', uk)$.

To check if the tuple given by A constitutes a successful forgery, the *WBRPE* $WB-UNF-PRG$ experiment first unhardens and validates the forgery $\left\langle y, \tilde{P}, \tilde{t}, K \right\rangle \leftarrow \mathcal{U}_{uk,vk}(\omega, \text{UP}_K, t')$ consequently performing the following test

$$\left\langle y, \tilde{P}, \tilde{t}, K \right\rangle \neq \perp \ \wedge \ ((uk \notin UK) \ \vee \ (\text{UP}_K \notin P[uk]))$$

Namely, either the adversary generated a successful forgery without obtaining the key uk or it obtained the key uk but for a different program UP_K. ∎

5 Universal WBRPE with Validation (WBRPEwV)

The *Universal WBRPEwV* scheme WV' with validation provides remote inputs confidentiality, in addition to confidentiality and integrity properties of local host inputs. The universal program is identical to the universal program UP, in Figure 15, of the *Universal WBRPE* scheme, in section 3, which implies that this extension of the scheme does not require more, than the protection of the original universal program UP, i.e. if we can protect the same universal program then we obtain a *Universal WBRPEwV* for any program, which in addition to the properties of confidentiality and integrity also provides privacy of the remote inputs.

The scheme WV' will perform inputs validation, by running the validation procedure V' on input programs P. In addition, the specific *WBRPEwV* scheme WV which underlies the construction of WV' will perform validation of the universal programs UP generated in the hardening procedure $\mathcal{HV}'$, in 32, by performing validation thereof with V.
Provided both the universal and the input program are valid, the originator will obtain the final result of the computation, i.e. $P_{t,l}(a)$ for some a. The universal program in Figure 15. During the generation phase of the *Universal WBRPEwV* scheme WV', with privacy of remote inputs, the generation procedure $\mathcal{GV}'$ obtains in an input the security parameter and a pair $V', t_{V'}$. It then generates the validation procedure V to validate the inputs of the specific *WBRPEwV* scheme WV, applies $\mathcal{GV}$ with V, t_V and generates the OVM for the specific *WBRPEwV* and the keys. The OVM has the validation procedure V inside. Next it writes the OVM' of the *Universal WBRPEwV* that will merely perform parsing and formatting of the input and will execute OVM on the formatted input. The validation of programs will be performed by the OVM. The obfuscated virtual machine OVM' of the *Universal WBRPEwV* scheme, is constructed in the generation

Program $\mathcal{G}'(1^k, V', t_{V'})$ {

$V \leftarrow createV(V', t_{V'})$

$t_V = t_{V'} + 4$

$(hk, vk, \text{OVM}) \xleftarrow{R} \mathcal{GV}(1^k, V, t_V)$

OVM' $\leftarrow createOVM'(\text{OVM})$

return $\langle hk, vk, \text{OVM'}\rangle$

}

Figure 29: Generation procedure of the *Universal WBRPEwV*.

$createV(\mathbf{V'}, \mathbf{t_{V'}}) =''$ $V(\text{UP}_d, \sigma_{UP})$ {

$(\tilde{K}, c_P) \leftarrow$ parse σ_{UP}

$(P, \sigma) \leftarrow \mathcal{VD}_K(c_P)$

if $(\mathbf{V'_{t_{V'},1}}(P, \sigma) = 1)$

if $(createUP(\tilde{K}) = \text{UP}_K)$ {$return$ 1}

return 0

}''

Figure 30: The validation program of the *Universal WBRPEwV*.

procedure using $createOVM'$, in Figure 31.

5.0.1 Hardening Procedure

The hardening procedure of the universal scheme obtains the input program and the validation parameter. It applies the generation procedure $\mathcal{G}_{AE}$ of the authenticated encryption and generates a pair of keys, next the input program is encrypted and authenticated along with the validation parameter. The universal program is generated for the corresponding secret key. At this stage, the hardening procedure $\mathcal{H}'$ generates a validation parameter of the universal program that will be used by the OVM of the specific *WBRPEwV*. The validation parameter of the universal program UP is comprised of the secret key embedded inside UP and of the encryption (computed earlier of the input program and its corresponding validation parameter) therefore binding the particular UP_K that will be passed to the OVM in an input along with the input program P and the key pair generated during this phase. In particular, this bond prevents an adversary from tampering with the string, by substituting the validation parameter and the input program with other

createOVM′(OVM) {

return "*read* (c, a, t, l)

(c$_{UP}$, c$_P$) ← parse c

a' ← (a, t, l, c$_P$)

t'=p(t)+4

l'=l+|P|+|t|+|K|

write (c$_{UP}$, a', t', l')″

‖OVM

}"

Figure 31: The *createOVM'* function, that generates the OVM'.

values. Then the validation parameter and the UP are hardened and the $\mathcal{HV}'$ outputs the encrypted input program, hardened universal program, the ephemeral unhardening key concatenated with the secret key.

Program $\mathcal{HV}'(P, \sigma, hk)$ {

$K \xleftarrow{R} \mathcal{G}_{AE}(1^k)$

$c_P \leftarrow \mathcal{AE}_K(P, \sigma)$

$\mathrm{UP}_K \leftarrow createUP(K)$

$\sigma_{UP} \leftarrow (K, c_P)$

$(c_{UP}, uk) \leftarrow \mathcal{HV}_{hk}(\mathrm{UP}_K, \sigma_{UP})$

$c \leftarrow \langle c_{UP}, c_P \rangle$

return $(c, \langle uk, K \rangle)$

}

Figure 32: The hardening procedure of the *Universal WBRPEwV*.

Program $\mathcal{UV}'_{uk,vk}(\omega, P, t)$ {

$\left\langle y, \tilde{P}, \tilde{t}, \tilde{K} \right\rangle \leftarrow \mathcal{UV}_{uk,vk}(\omega)$

if $((P, t) = NULL)$ { return y}

if $((\tilde{P}, \tilde{t}, \tilde{K}) \neq (P, t, K))$ { return $\perp$

$\mathrm{UP}_{K'} \leftarrow createUP(K')$

$t' = t + 3$

$\left\langle y, \tilde{P}, \tilde{t}, K \right\rangle \leftarrow \mathcal{UV}_{uk,vk}(\omega, UP_K, t')$

return y

}

Figure 33: The unhardening procedure of the *Universal WBRPEwV*.

5.0.2 The Unhardening Procedure

Identical to the unhardening procedure $\mathcal{U}'$ of the *Universal WBRPEwV* in section 3, Figure 19.

Lemma 5.1 *Let* $WV = (\mathcal{GV}, \mathcal{HV}, \mathcal{UV})$ *be a* $WB - PRV$ *secure specific* WBRPEwV *scheme* WV *for* UP *and let* $\Pi = (\mathcal{G}_{AE}, \mathcal{AE}, \mathcal{VD})$ *be a secure authenticated encryption scheme. Then* $WV' = (\mathcal{GV}', \mathcal{HV}', \mathcal{UV}')$ *is a* $WB - PRV$ *secure* WBRPEwV *scheme* $WV'y$ *for any program.*

.

Proof The proof of lemma follows from claim 5.2.

Claim 5.2 *For every PPT algorithm A and every program P, let $\mathtt{Adv}^{WB-PRV}_{WV',A'}(k)$ be the advantage that the algorithm A' gains during the $WB-PRV$ experiment against WV', then*

$$\mathtt{Adv}^{WB-PRV}_{WV,A}(k) \geq \mathtt{Adv}^{WB-PRV}_{WV',A'}(k)$$

Proof Given a PPT algorithm $A' = (A'_1, A'_2)$ against the *Universal WBRPEwV* scheme WV', construct a PPT algorithm $A = (A_1, A_2)$ against the *WBRPEwV* scheme WV the $\mathbf{Exp}^{WB-PRV}_{WV,A}(k)$ experiment.
A receives all the keys in an input and uses A' as a black box. We show that the view of A' when executed in the environment of $\mathbf{Exp}^{WB-PRV}_{WV',A'}(k)$ is distributed identically to its view when invoked by A in $\mathbf{Exp}^{WB-PRV}_{WV,A}(k)$. We briefly present the course of the $WB-PRV$ experiment of the *Universal WBRPEwV* scheme WV', and subsequently compare it to the execution simulated by A.

Algorithm $A_1^{REM_k}(1^k, \tau)$
- $(V', t_{V'}, s') \leftarrow A'^{REM_k}_1(1^k, \tau)$
- $V \leftarrow createV(V', t_{V'})$
- $t_V = t_{V'} + 4$
- $s \leftarrow \langle V', t_{V'}, \tau \rangle$
- return $(1^k, V, t_V, \langle s, s' \rangle)$

Algorithm $A_3^{REM_k}(\omega, \langle s, s' \rangle)$
- return $A'^{REM_k}_3(\omega, uk, s')$

Algorithm $A_2^{REM_k}(hk, \text{OVM}, vk, \langle s, s' \rangle)$
- $\langle V', t_{V'}, \tau \rangle \leftarrow$ *parse* s
- OVM' $\leftarrow createOVM'$(OVM)
- $(c, a_{ADV}, t, l, uk, s') \leftarrow A'^{REM_k}_2(\text{OVM'}, hk, vk, s)$
- $(c_{UP}, c_P) \leftarrow$ *parse* c
- $t' = t + 4$
- $l' = l + |P| + |t| + |K|$
- $a_{ADV} \leftarrow (a_{ADV}, t, l, c_P)$
- $s \leftarrow \langle s, \text{OVM'}, hk, vk \rangle$
- return $(c_{UP}, a_{ADV}, t', l', uk \langle s, s' \rangle)$

Figure 34: The implementation of $A = (A_1, A_2, A_3)$ in $\mathbf{Exp}^{WB-PRV}_{WV,A}(k)$.

The $\mathbf{Exp}^{WB-PRV}_{WV',A'}(k)$ experiment invokes A'_1 and obtains $(V', t_{V'}, s')$ back, where V' is the validation program, $t_{V'}$ is the number of steps to execute V' and s' is the state information. It subsequently constructs V and computes the number of steps t_V to execute V. It then generates the keys and the OVM and invokes A'_2 with (hk, vk, OVM, s) in an input, which in turn returns $(c, uk, a_{ADV}, t, l, s)$, where a_{ADV} is the part of the remote input which the adversary has control over.

The experiment runs OVM' on (c, a_{ADV}, t, l), obtains ω, invokes A'_3 with input ω and returns whatever A'_3 returns.

Now consider the simulation of A which runs in the environment of the $\mathbf{Exp}^{WB-PRV}_{WV,A}(k)$ experiment. Initially the $\mathbf{Exp}^{WB-PRV}_{WV,A}(k)$ experiment invokes A_1 with input that is comprised of the security parameter and the τ, which in turn invokes A'_1 and obtains $(V', t_{V'}, s')$ back. It then generates s, V, computes t_V and returns these to the experiment.
During the next stage, the experiment applies $\mathcal{G}$, generates the keys and the OVM and invokes A_2.

A_2 generates the OVM' for A'_2 according to the construction, and executes A'_2. At this stage it obtains c of the universal scheme along with other parameters returned by A and has to generate the remote input and the hardened program which will be compatible with the design of the experiment for WBRPE, i.e. the hardening procedure obtains universal programs as input and the experiment executes them in OVM and invokes A_3 with ω, i.e. the result of the execution. Since the result of the execution of P on a is identical to the result of UP_K on a', where $a' \leftarrow (a, a_{ADV}, t, l, c_P, V, t_V)$, A_3 will invoke A'_3 on this ω and will return its response back to its experiment. Given the function of the remote input of the *Universal WBRPEwV* scheme WV' output by A', the result computed by A is a function of the remote input of the *WBRPEwV* scheme WV, since whatever A' learns about the remote input a can essentially be applied to A and the remote input a' to *WBRPEwV* scheme. ∎

Lemma 5.3 *Let* $WV = (\mathcal{GV}, \mathcal{HV}, \mathcal{UV})$ *be a* $WB - IND - CPA - \varphi$ *secure specific* WBRPEwV *scheme* WV *for* UP *and let* $\Pi = (\mathcal{G}_{AE}, \mathcal{AE}, \mathcal{VD})$ *be a secure authenticated encryption scheme. Then* $WV' = (\mathcal{GV}', \mathcal{HV}', \mathcal{UV}')$ *is a* $WB - IND - CPA - \varphi$ *secure* WBRPEwV *scheme* WV' *for any program.*

Intuitively, since the σ parameter is hardened along with the input program P, this should not leak any information about P to the adversary. Next, the σ is only unhardened inside the OVM, which is assumed to be secure, therefore the indistinguishability specification is trivially obtained.

Lemma 5.4 *Let* $WV = (\mathcal{GV}, \mathcal{HV}, \mathcal{UV})$ *be a* $WB - UNF - OUT - \varphi$ *secure specific* WBRPEwV *scheme* WV *for* UP *and let* $\Pi = (\mathcal{G}_{AE}, \mathcal{AE}, \mathcal{VD})$ *be a secure authenticated encryption scheme. Then* $WV' = (\mathcal{GV}', \mathcal{HV}', \mathcal{UV}')$ *is a* $WB - UNF - OUT - \varphi$ *secure* WBRPEwV *scheme* WV' *for any program.*

Lemma 5.5 *Let* $WV = (\mathcal{GV}, \mathcal{HV}, \mathcal{UV})$ *be a* $WB - IND - CPA - \varphi$ *and a* $WB - UNF - PRG$ *secure specific* WBRPEwV *scheme* WV *for* UP *and let* $\Pi = (\mathcal{G}_{AE}, \mathcal{AE}, \mathcal{VD})$ *be a secure authenticated encryption scheme. Then* $WV' = (\mathcal{GV}', \mathcal{HV}', \mathcal{UV}')$ *is a* $WB - UNF - PRG$ *secure* WBRPEwV *scheme* WV' *for any program.*

If there is an adversary B' that can forge the output of a computation of program P then it is trivial to construct an adversary A' that will forge the output of P, when the input is P and σ. Furthermore, if B' can forge the tuples P, σ then we can construct A' that returns $P||\sigma$ such that σ is a forgery for WV'.

6 Applications of *WBRPE*

We present two selected applications, and show how these could be securely implemented with the *WBRPE* scheme. The implementations presented below are on a conceptual level and aimed at providing of the general feeling on the applicability of the *WBRPE* scheme.

6.1 Online Publicly Accessible Database

Generally applications based on the setting of online database, involve two parties, a server which holds the database and a client who wishes to query the database. The privacy and the integrity of both the local and the remote hosts should be provided. Much research was devoted to this issue, focusing on the protection of the query submitted by the user to the remote database, e.g. the model of *Private Information Retrieval* in [15], as well as on the protection of the database from malicious users, i.e. *Data Mining*, [16], [17]. Other solutions address both requirements [18].

We next present how to apply the *WBRPE* directly in order to map the security requirements of applications based on online databases.

In *WBRPE* scheme, the client is the local host and the server is the remote host. The input supplied by the client is a query, and the remote input of the server is a database, and the client wishes to compute the result of its query on the database.
The owner of the database defines the set of valid queries which can be performed on the database. More specifically, it generates the validation program V that will validate each submitted query. This is typical in applications where it is required to enable the involved parties to perform statistical analysis on the data without compromising the individual records of the database, e.g. [19], [20].

During the generation phase the hardening and the verification keys along with the obfuscated virtual machine are generated. The *OVM* has the embedded validation procedure, and is installed on the server.

The client generates the query, performs hardening and submits for execution on the server. The *OVM* upon input a hardened query, unhardens it with the corresponding unhardening key and validates the query, e.g. to make sure that the query result does not exceed some predefined size, [21] or that the query does not compromises the privacy of individual records, [22]. If the validation process succeeds then the *OVM* queries the database with that query, hardens the result and sends back to the originator. The originator upon receipt the hardened query, unhardens to obtain the final result of the computation. The client can

also perform the validation of the result, i.e. to make sure that the result is indeed the computation based on the query that it submitted. To accomplish this, it simply supplies the original query as an additional input to the unhardening procedure.

Clearly the privacy and the integrity of both inputs of the client and the server are obtained, since the server cannot observe the queries submitted by the client, further since the database is queried inside the OVM the server cannot observe the process of the computation. Furthermore, the scheme ensures that only valid queries can be executed, by defining the valid queries set prior to the generation process, s.t. each query will be validated according to that set.

6.2 Grid Computing

In a grid computing environment, a large number of users (nodes) donate their idle CPU cycles to perform computation on behalf of a third party (the client).In this work we focus on (distributed) computing tasks where the jobs require only CPU cycles (i.e., we exclude jobs that need user input/output or large scale storage). From a commercial perspective, cryptanalysis of keys seems to be a major application of such scenarios.

In a typical scenario based on this model, the participating nodes decide what and how much of their resources they are willing to devote. A job scheduler then distributes standalone programs (jobs) P to each of these nodes, along with an input a "hardwired". In most cases, P can run uninterrupted for days without requiring any input from the nodes (the only input needed is a). For simplicity, we consider a single centralized scheduler, although this model can be extended to multiple grids/schedulers. There are two main security concerns for the client.

1. (confidentiality) The job (P, a) may contain sensitive information that needs to be protected from the user.

2. (Integrity) A user may tamper with the execution of the job and/or submit incorrect results. That is, it may submit a result that is $\neq P(a)$.

A WBRPE satisfying IND and UNF-OUT may be used to satisfy both the above requirements. The scheduler embeds (P, a) into a program P' that simply computes $P(a)$ and sends $(OVM, H_{hk}(P')$ to the attacker. If the IND requirement for WBRPE is satisfied, then it is guaranteed that the attacker cannot learn much about (P, a) except the length. Similarly, if the UNF-OUT requirement is satisfied then it can be shown that the attacker cannot make the scheduler accept anything apart from $P(a)$ or $\perp$.

From the nodes' perspective, there are some security issues too- a job might contain malicious code, which the node is not willing to execute. This issue is addressed by the Validity requirement of WBRPE.

7 Related Works

Program obfuscation was formally defined by Barak *et al.* [1] based on black box simulation, which also proved that obfuscation is theoretically impossible for general purpose functions and for some specific functions. Following the definition presented by Barak *et al.*, Wee, [23], presented a positive result for point functions obfuscation. The definition presented by Barak *et al.* [1] was further extended to include auxiliary inputs by Goldwasser and Kalai in [24], which in addition proved that the construction presented by Wee [23] holds in their model. Furthermore, the prominent impossibility result of Barak et al. [1] also holds in their weaker model.

In contrast to the impossibility result of Barak *et al.* [1] and Goldwasser and Kalai, [24], there are other positive solutions, e.g. an NP-hardness result of Wang [25], Ogiso [26], a PSpace hardness result of Chow *et al.* [27]. There are also alternative weaker definitions of obfuscation, e.g. Hohenberger *et al.* [28], that present a positive obfuscation result for cryptographic re-encryption functionality, obfuscation for access control Lynn [29], also a weaker definition of Best Possible Obfuscation in Goldwasser em et al. [30], which makes a relaxed requirement that the obfuscated program leaks as little information as any other program

with the same functionality, present a separation between black box and best possible obfuscation, and show tasks which can be achieved under this new definition.

It is conjectured that obfuscation alone does not provide a solution to remote program execution concept, and should be used in tandem with other techniques to obtain provably secure protocols and frameworks for execution of programs in remote untrusted environments, and in particular it is not a substitution to these techniques.

Theoretical solutions also focus on mobile code cryptography (a.k.a. encrypted computation) that aims at providing black box security by employing provably secure cryptographic techniques. This approach allows a program to be executed securely on a remote untrusted host by transforming program's code into an encrypted form, consequently obtaining encrypted executable program that consists of instructions and operate on encrypted inputs, such that the remote host cannot inspect the original program.

Sander and Tschudin [31], [32], initiated mobile code research and were the first to identify the possibility of securely executing a code on a remote untrusted host, by proposing an application of encrypted computation techniques to the problem of protecting intellectual property, secret functions and mobile code from malicious hosts. They pointed out that protocols for secure multi-party computation could be useful in the design of a software only solution to protect mobile code against malicious hosts. Initially they found that polynomial functions can be encrypted for non-interactive evaluation if an algebraic homomorphic trapdoor one-way functions exist. Sander and Tschudin presented a non-interactive Computing with Encrypted Functions (CEF) protocol for computing polynomials and rational functions, using homomorphic encryption scheme. However their solution is limited to evaluation of polynomials and rational functions and is highly inefficient for practical use. Similar solutions were presented in [33], [34]. This approach can be further employed by function hiding as is extended by Sander and Tschudin in [35], which essentially means that the result of the computation is returned to the remote host upon receipt of the encrypted result by the originator. A subsequent work by *Sander et al* [36] gave a non-interactive CED protocol for all NC^1 functions. Based on that, a non-interactive CEF protocol can be implemented by letting Client's private input be its function f and server's function be a universal circuit. However due to the logarithmic limitation on the depth of the circuit being evaluated , its application is limited.

Although mobile code cryptography provides black box security, it is still difficult to achieve, and in particular, existing techniques exhibit various problems for practical applications, such as lack of efficiency, restriction to limited set of functions, and more.

In homomorphic encryption scheme for polynomial functions the possible number of terms in a function is exponential to the number of inputs, therefore the encrypted function to be transfered will be large. There is also a technical difficulty to find encryption schemes that can transform arbitrary functions to their encrypted executable versions.

Other approaches are based on Yao's secure function evaluation protocol [37] present solutions to protect program's code and data as well as host's data, and it is more powerful in terms of the type of functions it can compute.
Other works based on secure circuit evaluation in tandem with oblivious transfer, e.g. [38], [2], [39]. Unfortunately there are inherent disadvantages, e.g. Yao's [37] non-interactive protocol, in order to reduce interactiveness, leaks the whole circuit structure.

An additional common limitation of both secure distributed computation and encrypted computation schemes is the representation of programs as functions. When a program is represented by a function the encoding size may be exponential, thus increasing both computation and communication complexity.

8 Conclusions and Open Questions

In this work we focus on layering rigorous foundations of white box cryptography, by investigating the building blocks in white box security and presenting constructions, along with reductions to the underlying building blocks. At the moment there does not exist a proof showing that any white-box scheme is realizable, even by a reduction showing equivalence to a problem which is considered hard. On the other hand, there is no proof, even by reduction, that any of the white-box schemes is unrealizable. This motivates exploring

other, related, weaker or stronger notions of white-box security, to try to find some notion that we can prove realizable or unrealizable.

As we discussed in section 7, the existing results in mobile code are insufficient for practical applications, therefore we raise the question whether better results can be obtained with our white box primitive, the *WBRPE* scheme. In addition, the question whether there exists a *WBRPE* scheme for any program seems an interesting open problem.

Below outlined several open research directions which we find particularly important and interesting

- Currently the generation phase is performed by the trusted third party, and it is interesting to consider variations that would not rely on a strong trust assumption.
- Our scheme does not support state, which is crucial for computations performed by mobile agents, and is an essential extension to explore.
- Another interesting direction is the investigation of weaker white box primitives. In particular, to find the minimal assumption necessary to obtain security, i.e. the weakest building block, like a one way function in traditional cryptography.
- In parallel to theoretical research it is interesting to build a *WBRPE* candidate.

9 Acknowledgements

We thank Yoram Ofek, Jasvir Nagra and Christian S. Collberg for useful discussions helpful comments. This work was supported by funds from the European Commission (contract N 021186-2 for the RE-TRUST project).

References

[1] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, London, UK, 2001. Springer-Verlag.

[2] J. Algesheimer, C. Cachin, J. Camenisch, and G. Karjoth. Cryptographic security for mobile code. In *SP '01: Proceedings of the 2001 IEEE Symposium on Security and Privacy*, page 2, Washington, DC, USA, 2001. IEEE Computer Society.

[3] M. Bellare, J. Kilian, and P. Rogaway. The Security of the Cipher Block Chaining Message Authentication Code. *Journal of Computer and System Sciences*, 61(3):362–399, 2000.

[4] J. Daemen and V. Rijmen. *The Design of Rijndael: AES–the Advanced Encryption Standard*. Springer, 2002.

[5] D.E.S.E. Standard. National Bureau of Standards (US). *Federal Information Processing Standards Publication*, 46.

[6] H. Wee. On obfuscating point functions. In *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 523–532, New York, NY, USA, 2005. ACM Press.

[7] C.S. Collberg and C. Thomborson. Watermarking, tamper-proofing, and obfuscation-tools for software protection. *Software Engineering, IEEE Transactions on*, 28(8):735–746, 2002.

[8] S. Chow, P. A. Eisen, H. Johnson, and P. C. van Oorschot. White-box cryptography and an AES implementation. In *SAC '02: Revised Papers from the 9th Annual International Workshop on Selected Areas in Cryptography*, pages 250–270, London, UK, 2003. Springer-Verlag.

[9] S. Chow, P. A. Eisen, H. Johnson, and P. C. van Oorschot. A white-box DES implementation for DRM applications. In J. Feigenbaum, editor, *Digital Rights Management Workshop*, volume 2696 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2002.

[10] O. Billet, H. Gilbert, and C. Ech-Chatbi. Cryptanalysis of a white box AES implementation. *Selected Areas in Cryptography*, 3357:227–240.

[11] L. Goubin, J.M. Masereel, and M. Quisquater. Cryptanalysis of white box DES implementations. *Proceedings of the 14th Annual Workshop on Selected Areas in Cryptography*, 2007.

[12] Freedman, Ishai, Pinkas, and Reingold. Keyword search and oblivious pseudorandom functions. In *Theory of Cryptography Conference (TCC), LNCS*, volume 2, 2005.

[13] O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press New York, NY, USA, 2004.

[14] M. Bellare, P. Rogaway, and D. Wagner. A conventional authenticated-encryption mode, 2003.

[15] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private Information Retrieval. *Journal of the ACM*, 45(6):965–982, 1998.

[16] Verykios, Bertino, Fovino, Provenza, Saygin, and Theodoridis. State of the art in privacy preserving data mining. *SIGMODREC: ACM SIGMOD Record*, 33, 2004.

[17] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pages 439–450. ACM Press, May 2000.

[18] Gertner, Ishai, Kushilevitz, and Malkin. Protecting data privacy in private information retrieval schemes. *JCSS: Journal of Computer and System Sciences*, 60, 2000.

[19] N.R. Adam and J.C. Worthmann. Security-control methods for statistical databases: a comparative study. *ACM Computing Surveys (CSUR)*, 21(4):515–556, 1989.

[20] W. Du, Y.S. Han, and S. Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. *Proceedings of the 4th SIAM International Conference on Data Mining*, 233, 2004.

[21] D.E. Denning, P.J. Denning, and M.D. Schwartz. The tracker: a threat to statistical database security. *ACM Transactions on Database Systems (TODS)*, 4(1):76–96, 1979.

[22] F. Y. Chin and G. Özsoyoglu. Auditing and inference control in statistical databases. *IEEE Transactions on Software Engineering*, 8(6):574–582, November 1982.

[23] Wee. On obfuscating point functions. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 2005.

[24] S. Goldwasser and Y.T. Kalai. On the impossibility of obfuscation with auxiliary input. *FOCS*, 5:553–562, 2005.

[25] C. Wang. *A Security Architecture for Survivability Mechanisms*. PhD thesis, University of Virginia.

[26] T. Ogiso, Y. Sakabe, M. Soshi, and A. Miyaji. Software Tamper Resistance Based on the Difficulty of Interprocedural Analysis. *3rd Workshop on Information Security Applications (WISA 2002), Korea, August*, 2002.

[27] S. Chow, Y. Gu, H. Johnson, and V.A. Zakharov. An Approach to the Obfuscation of Control-Flow of Sequential Computer Programs. *Information Security: 4th International Conference, ISC 2001, Malaga, Spain, October 1-3, 2001: Proceedings*, 2001.

[28] S. Hohenberger and G.N. Rothblum. Securely Obfuscating Re-Encryption. *TCC*, pages 233–252, 2007.

[29] B. Lynn, M. Prabhakaran, and A. Sahai. Positive Results and Techniques for Obfuscation. *Advances in cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004: Proceedings*, 2004.

[30] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. In Salil P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 194–213. Springer, 2007.

[31] T. Sander and C.F. Tschudin. Towards mobile cryptography. In *RSP: 19th IEEE Computer Society Symposium on Research in Security and Privacy*, 1998.

[32] T. Sander and C. Tschudin. Protecting mobile agents against malicious hosts. In G. Vigna, editor, *Mobile Agents and Security*, volume 1419 of *Lecture Notes in Computer Science*, pages 44–60. Springer-Verlag, Berlin Germany, 1998.

[33] H. Lee, J. Alves-Foss, and S. Harrison. The use of encrypted functions for mobile agent security. *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, pages 297–306, 2004.

[34] P. Kotzanikolaou, M. Burmester, and V. Chrissikopoulos. Secure Transactions with Mobile Agents in Hostile Environments. *Information Security and Privacy: 5th Australasian Conference, ACISP'2000, Brisbane, Australia, July 10-12, 2000: Proceedings*, 2000.

[35] T. Sander and C.F. Tschudin. On software protection via function hiding. *Information Hiding*, pages 111–123, 1998.

[36] T. Sander, A. Young, and M. Yung. Non-interactive cryptocomputing for NC 1. In *IEEE Symposium on Foundations of Computer Science*, pages 554–567, 1999.

[37] A.C. Yao. How to generate and exchange secrets. *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.

[38] C. Cachin, J. Camenisch, J. Kilian, and J. Muller. One-round secure computation and secure autonomous mobile agents. In *Automata, Languages and Programming*, pages 512–523, 2000.

[39] Zhong and Yang. Verifiable distributed oblivious transfer and mobile agent security. In *DialM: Proceedings of the Discrete Algorithms and Methods for Mobile Computing & Communications; later DIALM-POMC Joint Workshop on Foundations of Mobile Computing*, 2003.