



Sixth Framework Programme
Information Society Technology



RE-TRUST

Remote EnTrusting by RUn-time Software auThentication

Project Number: **FP6 - 021186**

Deliverable: **D3.3**

Encrypted Code Final Report

1. Summary

Project Number:	FP6-021186
Project Title:	RE-TRUST: Remote EnTrusting by RUn-time Software auThentication
Deliverable Type:	R
Deliverable Number:	D3.3
Contractual Date of Delivery:	August 2009
Actual Date of Delivery:	August 2009
Title	Encrypted Code Final Report
Workpackage Contributing to the Deliverable:	WP3
Nature of the Deliverable:	Report
Author(s):	Brecht Wyseur, and Mina Deng (K.U.Leuven)
Reviewer(s):	
Abstract:	This deliverable presents an overview of the study that was conducted in task T3.3. The objective was to go beyond techniques for obfuscation, and investigate the concepts of Computing on Encrypted Data (CED) and Computing with Encrypted Functions (CEF). We present these concepts, discuss their state-of-the-art, and how these techniques can be useful for the RE-TRUST project.
Keywords:	Computing on Encrypted Data, Computing with Encrypted Functions, homomorphic functions, garbled circuits, secure function evaluation, white-box remote program execution
Classification:	Public

Contents

1	Introduction	4
1.1	Motivation	4
1.2	Contribution and outline	4
2	Computing on Encrypted Data	6
2.1	Notation	8
2.2	DCRA-based homomorphic cryptosystems	9
2.2.1	Goldwasser-Micali cryptosystem	11
2.2.2	Benaloh Cryptosystem	12
2.2.3	Naccache-Stern cryptosystem	14
2.2.4	Okamoto-Uchiyama cryptosystem	14
2.2.5	Paillier cryptosystem	16
2.2.6	Damgård-Jurik cryptosystem	18
2.3	Pairing-based extensions	20
2.3.1	Evaluating 2-DNF formulas on Ciphertexts	21
2.4	ElGamal class of homomorphic encryption schemes	22
2.4.1	Cramer-Shoup cryptosystem	23
2.4.2	ElGamal-Paillier amalgam	25
2.5	Lattice-based full homomorphic encryption	26
3	Computing with Encrypted Functions	28
3.1	Secure Function Evaluation	28
3.2	Universal Circuits	29
4	Towards RE-TRUST solutions based on CED/CEF	30
4.1	White-Box Remote Program Execution	30
	Bibliography	32

1 Introduction

This deliverable captures the work that was performed within the context of Task 3.3 of the RE-TRUST project. We will discuss the study that was conducted on ‘computing in the encrypted domain’, and how this can be used to address the remote entrusting paradigm.

1.1 Motivation

The RE-TRUST architecture [11, 40] foresees technology to perform computations on an untrusted machine, given a continuous network connection between the untrusted machine, and trustworthy entities.

The obfuscation techniques that have been developed within the context of this project deter an adversary from analysis and tampering of the application that is deployed. These are practical techniques that ‘raise the bar’ against attacks from the untrusted computing platform. Their main purpose is to extend the time frame in which we believe the protection techniques are still valid, before a new update needs to be released. With respect to this, a Security Analysis [2] has been performed, which includes a study of metrics for obfuscation techniques and an empirical study.

The objective of Task 3.3 of the RE-TRUST project is to go beyond the obfuscation techniques that have been developed within the context of Task 2.4 of the project, and search for provable secure solutions. Since an adversary can monitor all memory pages, CPU calls, and has direct access to the implementation of the software that is executed on the untrusted computing device, novel techniques are required to protect access to the data and the program flow. In this task, we identify two main concepts: 1) computing on encrypted data, and 2) computing with encrypted functions.

Computing on Encrypted Data (CED). To protect application data from analysis, we naturally resort to cryptographic functions that provide confidentiality of data. Instead of storing and computing with the original data, we store the data in encrypted form and compute on the encrypted data. This however requires special functions that are able to evaluate encrypted data without decrypting them (intermediate decryption on the untrusted platform might be intercepted by the adversary).

Computing with Encrypted Functions (CEF). The dual case is where we want to hide the semantics of the functions that we compute on the untrusted platform.

We denote with ‘computing in the encrypted domain’ as computations that are both performed on encrypted data as with encrypted functions.

1.2 Contribution and outline

The activities of this task mainly focussed on the study of the state of the art in techniques to compute in the encrypted domain, and how the existing techniques can

be applied to support solutions for the remote entrusting paradigm.

The development of new techniques is beyond the scope of this task. A large community of cryptographers and mathematicians is searching for new techniques. Competing with such a community as a small task within the context of this project is a utopia. It would require a much more extensive study in the fundamentals of the underlying mathematical structure and long term commitment by several experienced researchers.

Nevertheless, with the limited resources that we were able to dedicate to this task, we have performed an in-depth study of the state of the art in homomorphic cryptosystems which we describe in Section 2. These cryptosystems enable computations on encrypted data, without the need to decrypt this data. In Section 3, we give an overview of fundamentals for techniques that enable to compute with encrypted functions.

The knowledge obtained from these studies has been used to investigate solutions for the ‘White-Box Remote Program Execution’ paradigm, an architecture that has been developed for the RE-TRUST project as a framework for provably secure solutions. This is described in Section 4.1.

2 Computing on Encrypted Data

The RE-TRUST architecture describes a framework where software is executed on an untrusted platform. This software might contain sensitive data, whose confidentiality needs to be preserved. To protect the data, a natural solution would be to store that data in (cryptographically) encrypted form. The question remains, how this data can then still be used without revealing any information thereof. In this section, we will discuss techniques to perform computations on encrypted data, without intermediate decryption.

Suppose the software has two variables m_1 and m_2 , whose sum $(m_1 + m_2)$ needs to be computed on the untrusted platform. To preserve the confidentiality, the variables $c_1 = E(m_1), c_2 = E(m_2)$ are stored while the value $E(m_1 + m_2)$ needs to be computed instead. E denotes a cryptographic encryption scheme. An obvious solution would be to perform the computation

$$E(m_1 + m_2) = E(D(c_1) + D(c_2)),$$

where D is the decryption function corresponding to E . However, when performing this computation on the untrusted platform, the value $D(c_1) = m_1$ needs to be stored in memory, which is accessible by the adversary.

One direction to address this issue, is the use of *homomorphic encryption schemes*, defined in Definition 1. Figure 1 depicts a generic architecture for homomorphic encryption schemes, where on the left hand side, a party has access to both an encryption and corresponding decryption operation, and (secret) inputs x_1, \dots, x_n . The function F is a public function that needs to be evaluated with x_1, \dots, x_n . On the right hand side are depicted the corresponding encrypted inputs, while G represents a function that corresponds to (and can be computed from) the function F (and is often denoted as $E(F)$). The ‘special property’ of homomorphic functions (defined in Definition 1, is that from the evaluation of G with the encrypted inputs, a party that has access to the decryption functionality can deduce the evaluation of F with the original inputs. The resemblance to the RE-TRUST architecture is quite clear in this paragraph: the trusted entities are those who have access to the decryption oracle, while the untrusted host corresponds to the party that evaluates G .

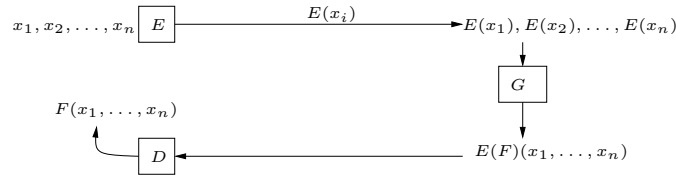


Figure 1: Computing in the Encrypted Domain

DEFINITION 1 (HOMOMORPHIC ENCRYPTION SCHEME) An encryption scheme is said to be *homomorphic* if for any given encryption key k , the encryption function E satisfies

$$\forall m_1, m_2 \in \mathcal{M} : E(m_1 \odot_{\mathcal{M}} m_2) = E(m_1) \odot_{\mathcal{C}} E(m_2)$$

for some operators $\odot_{\mathcal{M}}$ in \mathcal{M} and $\odot_{\mathcal{C}}$ in \mathcal{C} . \square

In the remainder of this section, we will present an overview of the schemes that we have studied. But first, we need to explain some relevant concepts.

Semantic security. A widely accepted notion of security for asymmetric encryption schemes is *semantic security*. For a (probabilistic) encryption scheme to be semantically secure, it must be infeasible for a computationally bounded adversary to derive additional *information* about a plaintext, when given the corresponding ciphertext (and the public key K). This notion was introduced by Goldwasser and Micali [22]. Subsequently, Goldwasser and Micali [23] showed that semantic security follows from *ciphertext indistinguishability*. This allowed the use of the IND-CPA security notion as a more commonly used notion to assess the security of asymmetric encryption schemes.

The IND-CPA security notion for a public-key cryptosystem defines that the adversary has access to the encryption oracle and needs to guess a secret bit. Hence, a public-key cryptosystem is defined to be IND-CPA secure if no adversary with access to an encryption oracle can pick two messages, of equal length, such that it can distinguish (still having encryption-oracle access) between the encryptions of the two. This notion is similar to the *find-then-guess* CPA (FTG-CPA) security by Bellare *et al.* [3] (for symmetric ciphers).

Semantic security is essential for homomorphic encryption schemes, since an adversary should not be able to distinguish the encryptions of commonly used values, such as **TRUE** (1) and **FALSE** (0), or distinguish between different usages of similar values (to avoid ‘linkability’).

RSA. An encryption scheme that is commonly presented as an example of a homomorphic encryption scheme, is RSA. It is an asymmetric encryption scheme, defined as

encryption: $c = m^e \mod n$

decryption: $m = c^d \mod n$,

where $n = p \cdot q$ is the product of two large primes p and q , and e, d are the respective public encryption and private decryption key such that $e \cdot d \mod (p-1)(q-1) \equiv 1$. Due to the distributivity of multiplication over exponentiation, RSA has homomorphic properties:

$$\begin{aligned} E(m_1) \cdot E(m_2) &\equiv (m_1^e \mod n) \cdot (m_2^e \mod n) \\ &\equiv (m_1^e \cdot m_2^e) \mod n \\ &\equiv (m_1 \cdot m_2)^e \mod n \\ &\equiv E(m_1 \cdot m_2). \end{aligned}$$

However, RSA does not satisfy semantic security, since an adversary can easily distinguish the encryption $E(1)$ since he can compute it himself given the public encryption key. To defeat this weakness provided by ‘chosen ciphertext attacks’ (CCA), researchers have introduced improved schemes, such as the RSA Optimal Asymmetric Encryption Padding (RSA-OAEP) [4] which have been proven secure under these types of attacks. Unfortunately, these improved schemes do not have the homomorphic properties any more.

The holy grail is the design of *ring/algebraic homomorphisms*. I.e., homomorphic encryption schemes that satisfy homomorphic relation between both the algebraic operators of the rings $(\mathcal{M}, +_{\mathcal{M}}, \times_{\mathcal{M}})$ and $(\mathcal{C}, +_{\mathcal{C}}, \times_{\mathcal{C}})$.

Outline. Homomorphic encryption schemes come in very different flavors, each built onto different mathematical structures and as a result with different properties. In this section, we will give an overview of the main classes of homomorphic encryption schemes, which include

- **DCRA-based.** Homomorphic public key encryption schemes that are based on the *decisional composite residuosity assumption* (DCRA).
- **ElGamal.** Encryption schemes that are based on the Diffie-Hellman key agreement. Their security relates to the hardness problem of computing discrete logarithms.
- **Lattice-based.** A rather new area in cryptography is lattice-based cryptography. Based on the hardness problem of computing the shortest vector in a lattice, public key encryption schemes with homomorphic properties have been presented.

Schemes based on other mathematical hardness assumptions, such as solving equations have been presented [15]. Studying all homomorphic encryption schemes was not the objective of task 3.3 in the RE-TRUST project, rather to learn from an interesting set of homomorphic encryption schemes, and find out how they can be useful for practical (while provable) results to address the remote entrusting paradigm.

2.1 Notation

Below, a brief overview of some essential notations that are often used in the remainder of this document.

E	A deterministic encryption
$r \leftarrow^R \mathbb{Z}$	An element r taken at random from the set \mathbb{Z}
\mathcal{E}_r	A probabilistic encryption, with randomness r
$\phi(n)$	The Euler ϕ evaluation of n (= order of the group \mathbb{Z}_n)
\oplus	Exclusive binary addition
$O(\cdot)$	The “Big O” operator to denote complexity

2.2 DCRA-based homomorphic cryptosystems

The *decisional composite residuosity assumption* (DCRA) is a mathematical hardness assumption that is the basis of a first class of important homomorphic encryption schemes. It was first proposed by Goldwasser and Micali [21] in 1982, and all the homomorphic encryption schemes can be proven secure by reduction to this intractability assumption.

In general, we can depict any of the encryption schemes in this family as follows. Consider a DCRA-based probabilistic encryption function \mathcal{E} .

$$\mathcal{E} : \mathbb{G} \rightarrow \tilde{\mathbb{G}},$$

where $\tilde{\mathbb{G}} \cong \frac{\mathbb{G}}{H}$ is the factor group of \mathbb{G} with $H \subset \mathbb{G}$, as depicted in Figure 2.3.1.

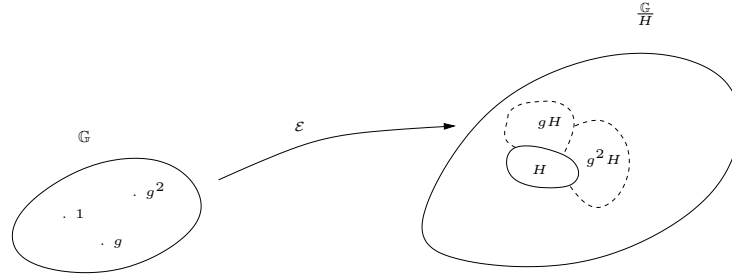


Figure 2: Quotient group mapping

\mathbb{G} represents the ‘plaintext’ domain, while $\tilde{\mathbb{G}}$ represents the encrypted domain. Multiplications that are performed in the encrypted domain correspond to addition operations in the plaintext domain due to the homomorphic property of the encryption scheme. The semantic security corresponds to the fact that an adversary should not be able to distinguish if two elements in the factor group belong to the same subgroup $g^i H$.

Mathematical basis. The GM cryptosystem is based upon the assumed intractability of the quadratic residuosity problem modulo a composite $n = pq$, where p, q are large primes. This is a special case of the general DCRA hardness assumption.

DEFINITION 2 (QUADRATIC RESIDUOSITY PROBLEM) For a given (x, n) , with $n = pq$, p, q large primes, it is difficult to determine whether x is a quadratic residue (QR) module n . I.e.,

$$\exists y \vdash x = y^2 \pmod n$$

□

Under the condition that p or $q \equiv 3 \pmod 4$, the quadratic residuosity problem leads to a homomorphic encryption scheme because of the following two properties:

Inversion. Under the condition that p or $q \equiv 3 \pmod{4}$, it holds that: if a is a QR \pmod{n} , then $(-1) \cdot a \pmod{n}$ is a QNR (quadratic non-residue) \pmod{n} (and vice versa).

Multiplication. The true-tables for multiplication of quadratic (non) residues module n has a similar structure as addition modulo 2.

a	b	$a \cdot b \pmod{n}$	a	b	$a \oplus b$
QR	QR	QR	0	0	0
QR	QNR	QNR	0	1	1
QNR	QR	QNR	1	0	1
QNR	QNR	QR	1	1	0

Observe that as a result,

$$\frac{QR}{QNR}, \cdot \cong \mathbb{Z}_2, +$$

which is the property that was initially used in the cryptosystem introduced by Goldwasser and Micali [21] (See Sect. 2.2.1). In the past, quadratic residues have been studied intensively, and several concepts were introduced to be able to compute with them and derive new properties. One of these concepts is the Legendre Symbol, which denotes whether or not a is a quadratic residue in a group \mathbb{Z}_p^* , with p prime, and is defined as follows:

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{if } \exists y \vdash a = y^2 \pmod{p} \\ 0 & \text{if } p|a \\ -1 & \text{else} \end{cases}$$

Using the formula's describe below, we can easily compute with the Legendre symbol.

- Factorize the nominator: $\left(\frac{\prod_i a_i}{p}\right) = \prod_i \left(\frac{a_i}{p}\right)$
- Compute the Legendre symbol: $\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod{p}$
- The reciprocity theorem, one of the most beautiful theorems in mathematics: $\left(\frac{p}{q}\right) \left(\frac{q}{p}\right) = (-1)^{\frac{(p-1)(q-1)}{4}}$.

The Jacobi Symbol is a generalization of the Legendre symbol for non-prime denominators, and defined as

$$\left(\frac{a}{\prod_i p_i^{\alpha_i}}\right) = \prod_i \left(\frac{a}{p_i}\right)^{\alpha_i}$$

Similar to the Legendre symbol, if $\left(\frac{a}{n}\right) = -1$ then a is not a quadratic residue of n because a was not a quadratic residue of some p_i . If a is a quadratic residue of n , $\left(\frac{a}{n}\right) = 1$. Unfortunately, the opposite cannot be said. If $\left(\frac{a}{n}\right) = 1$, a may or may not be a quadratic residue of n . This is because for a to be a residue (\pmod{n}) it has to be a

residue modulo every prime that divides n , but the Jacobi symbol will equal to 1 if a is a non-residue modulo zero, two (or any even number) of the primes dividing n .

The asymmetry of the system comes from the fact that one can generate new quadratic residues in \mathbb{Z}_n^* , even without the knowledge of the factorization of n . On the other hand, it is difficult to decide whether or not a number is a quadratic residue or not, when the factorization is unknown for composite groups \mathbb{Z}_n , where $n = pq$ is the product of two large distinct primes.

RSA leaks the Jacobi Symbol. This is exactly where RSA fails– it leaks the Jacobi Symbol– hence it cannot be used as a homomorphic encryption scheme in this way. Consider the RSA encryption of the message $m : c = E(m) = m^e \bmod n$. Then m is QR $\Leftrightarrow E(m)$ QR.

The quadratic residuosity assumption can be generalized as the decisional composite residuosity assumption over the group \mathbb{G} as follows.

DEFINITION 3 (DECISIONAL COMPOSITE RESIDUOSITY ASSUMPTION (DCRA)) Let A be a probabilistic polynomial time algorithm. Assume A knows the order of \mathbb{G} , which is k bits. A gets input x and outputs bit $b = 1$ if x is an n 'th residue in \mathbb{G} , $b = 0$ otherwise. Let $p(A, k, x)$ be the probability that $b = 1$. Then

$$|p(A, k, x) - p(A, k, x^n)| < \text{neg}(k)$$

□

When $\text{GCD}(n, \#\mathbb{G}) > 1$, then the n 'th residuosity decision is believed to be intractable.

2.2.1 Goldwasser-Micali cryptosystem

In the Goldwasser-Micali cryptosystem [21], $\tilde{\mathbb{G}} = \mathbb{G} \times H$, with $\mathbb{G} = \{0, 1\}$, and $H = \{x \mid x \text{ QR in } \mathbb{Z}_n^*\}$.

$$\mathcal{E} : \{0, 1\} \rightarrow \{0, 1\}.r$$

with $r \in H = \{x \mid x \text{ QR}\}$.

The cryptosystem If the public key is the modulus n , and quadratic non-residue x , then the encryption of a bit b is $\mathcal{E}(b) = r^2 x^b \bmod n$. Then we have the homomorphic property

$$\mathcal{E}(b_1) \cdot \mathcal{E}(b_2) = r_1^2 x^{b_1} r_2^2 x^{b_2} = (r_1 r_2)^2 x^{b_1 + b_2} = \mathcal{E}(b_1 \oplus b_2)$$

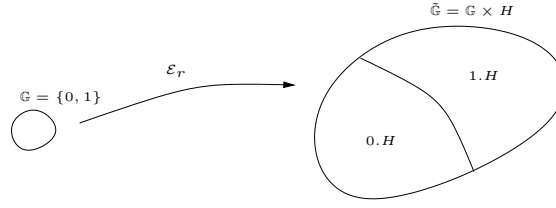


Figure 3: GM mapping

where \oplus denotes addition modulo 2, (i.e. exclusive-or).

Goldwasser-Micali is a probabilistic homomorphic single bit encryption scheme. It's very inefficient (expansion ratio $l(n)$), but due to its historical importance (most of the other homomorphic schemes that are used in practice, are based upon this one), it's mentioned here.

Prerequisites b is a single bit, $r \leftarrow^R \mathbb{Z}_n^*$ is a random element (\rightarrow random oracle model [33]), g is a quadratic non-residue, $n = pq$, where p and q are prime.

Probabilistic encryption

$$c = \mathcal{E}_r(b) = g^b r^2 \mod n$$

Deterministic decryption Decide whether or not the ciphertext is a quadratic residue with respect to n . This is only computable when the factorization of n is given.

2.2.2 Benaloh Cryptosystem

The Benaloh Cryptosystem [5] is an extension of the Goldwasser-Micali (GM) cryptosystem. The main improvement of the Benaloh Cryptosystem over GM is that longer blocks of data can be encrypted at once, whereas in GM each bit is encrypted individually. It also generalizes quadratic residuosity to r -th residuosity.

If the public key is the modulus n , the base g and the blocksize r , then the encryption of a message x is $g^x u^r \mod n$. Then we have the homomorphic property

$$\begin{aligned} \mathcal{E}_r(x_1) \cdot \mathcal{E}_r(x_2) \mod n &= (g^{x_1} u_1^r)(g^{x_2} u_2^r) \\ &= g^{x_1+x_2} (u_1 u_2)^r \\ &= \mathcal{E}(x_1 + x_2 \mod \phi(n)/r) \end{aligned}$$

The Cryptosystem

Prerequisites Choose a blocksize r , and two large primes p and q , such that $r|(p-1)$, and $\gcd(q-1, r) = 1$. Set $n = pq$, and choose $y \in (\mathbb{Z}/n\mathbb{Z})^*$ such that $y^{(p-1)(q-1)/r} \not\equiv 1 \pmod n$.

The public key is then g, n , and the private key is the two primes p, q .

This scheme works in the group $(\mathbb{Z}/n\mathbb{Z})^*$ where n is a product of two large primes.

Encryption To encrypt a message m , where $m \in \mathbb{Z}/r\mathbb{Z}$.

Choose a random $u \in (\mathbb{Z}/n\mathbb{Z})^*$

$$c = \mathcal{E}_r(m) = g^m u^r \pmod n$$

Decryption Decryption in the Benaloh cryptosystem, is rather a verification process. Just like in the Goldwasser-Micali cryptosystem, it is verified whether a given ciphertext is a r -th residue or not. For Benaloh, the basic observation is the following.

Raising the ciphertext c to the power of $(p-1)(q-1)/r$ gives the following result:

$$c^{(p-1)(q-1)/r} \equiv (g^m u^r)^{(p-1)(q-1)/r} \equiv g^{\frac{m}{r}(p-1)(q-1)} \pmod n$$

With the property that

$$c^{\phi(n)/r} \equiv 1 \pmod n \Leftrightarrow m \equiv 0 \pmod r$$

Because for a generator g , $\phi(n)$ is its order (i.e., the smallest power with which g can be raised to become 1). Hence it can be verified if the plaintext was 0 or not. If not, the exact value can be found by iterating over all possible i ($0 \leq i < r$), until

$$c^{i\phi(n)/r} \equiv 1 \pmod n \Leftrightarrow m \equiv -i \pmod r$$

because of the homomorphic property $E(m)E(i) = E(m+i \pmod r) = E(0 \pmod r)$.

However, for ‘large’ block size, decryption is quite expensive with such a brute force method. To improve this, for each equivalence class, a number T_M can be pre-computed.

$$T_M = g^{M(p-1)(q-1)/r} \pmod n$$

Then, for each $z \in E_r(M) : z^{(p-1)(q-1)/r} \equiv T_M \pmod n$.

This however, implies an intensive pre-computation step, and significant amount of storage. This can be improved by a **baby step-giant step** method. For each $M \approx k\sqrt{r}$ with k from 0 to \sqrt{r} , compute T_M (pre-computation phase). Then, for each i $0 \leq i \leq \sqrt{r}$, verify if $c^{i(p-1)(q-1)/r}$ is equivalent to one of the precomputed T_M . Then, $m = M - i \pmod r$.

The decryption complexity of Benaloh, with the baby step-giant step, is $O(\sqrt{r}l(r))$. Hence, Benaloh is only efficient for small blocksize. The expansion is $l(n)/l(r)$.

2.2.3 Naccache-Stern cryptosystem

Naccache-Stern homomorphic cryptosystem [29] is an improvement on the Benaloh cryptosystem, based on the hardness of computing higher residues modulo a composite RSA integer, to resolve the problem of r which has to be small for the decryption to be efficient. The encryption phase is analogously as Benaloh (and hence its expansion factor), and the expansion is still equal to $l(n)/l(k)$. The decryption complexity however is of magnitude $O(l^5(n) \log(l(n)))$, which is much more efficient when r becomes large, and the authors claim it is reasonable to choose the parameters as to get an expansion equal to 4.

Prerequisites The main difference with Benaloh is that the setup is slightly different. Let r be a B -smooth squarefree number. I.e., $r = \prod_i p_i$, with $p_i < B$, and all p_i different. Also, g does not need to be a generator, but its order does need to be a large order multiplicative of r .

Let σ be a squarefree odd B -smooth integer, where B is small integer and let $n = pq$ be an RSA modulus such that σ divides $\phi(n)$ and is prime to $\phi(n)/\sigma$. Generation of the modulus appears rather straightforward: pick a family p_i of k small odd distinct primes, with k even. Set $u = \prod_{i=1}^{k/2} p_i$, $v = \prod_{i=k/2+1}^k p_i$ and $\sigma = uv = \prod_{i=1}^k p_i$. Pick two large primes a and b such that both $p = 2au + 1$ and $q = 2bv + 1$ are prime and let $n = pq$. To select g , one can choose it at random and check whether or it has order $\phi(n) = 4$.

The public key is the numbers σ , n , g and the private key is the pair p , q . When $k = 1$ this is essentially the Benaloh cryptosystem.

Encryption Similar to Benaloh:

$$\mathcal{E}_r(m) = g^m u^r \mod n$$

Decryption The decryption is based on the Chinese remainder theorem, following an idea which goes back to the Pohlig-Hellman paper [35].

For each p_i that is a factor of r , $c^{\phi(n)/p_i}$ is compared with the pre-computed values $g^{i\phi(n)/p_i}$. If they are equal, then, $m \equiv i \mod p_i$. Because all p_i are smaller than B , this is efficient. Via Chinese remainder theorem, $m \mod \prod_i p_i$ is computed.

2.2.4 Okamoto-Uchiyama cryptosystem

Okamoto-Uchiyama cryptosystem [31] improves previous schemes, with a change to the base group G . Considering $n = p^2q$, p and q still being two large primes, the group $G = \mathbb{Z}_{p^2}^*$, it can be achieved as $k = p$. Therefore, the expansion is equal to 3. Its

advantage lies in the proof that its security is equivalent to the factorization of n . However, a chosen-ciphertext attack has been proposed leading to this factorization.

This scheme was used to design the EPOC systems. EPOC (Efficient Probabilistic Public Key Encryption) is a probabilistic public-key encryption scheme, developed in 1999 by T. Okamoto, S. Uchiyama and E. Fujisaki, is based on the random oracle model, in which a primitive public-key encryption function is converted to a secure encryption scheme by use of a truly random hash function; the resulting scheme is designed to be semantically secure against a chosen ciphertext attack. EPOC's primitive encryption function is the OU (Okamoto-Uchiyama) function, in which to invert the OU function is proven to be as hard as factoring a composite integer public-key. There are three versions of EPOC. EPOC-1 is designed for key-distribution; EPOC-2 and EPOC-3 are designed for both key-distribution and encrypted data transfer.

- EPOC-1 uses a one-way trapdoor function and a random function (hash function);
- EPOC-2 uses a one-way trapdoor function, two random functions (hash functions) and a symmetric-key encryption (e.g., one-time padding and block-ciphers);
- EPOC-3 uses the Okamoto-Uchiyama one-way trapdoor function and two random functions (hash functions) as well as any symmetric encryption scheme such as the one-time pad, or any classical block-cipher.

Note that earlier versions of EPOC were subject to security flaws as pointed out in [25].

The Okamoto-Uchiyama cryptosystem is a precursor to the Paillier cryptosystem, but has mostly been replaced by Paillier's system, which will be explained below.

Prerequisites

- Generate large primes p, q , and set $n = p^2q$.
- Choose $g \in (\frac{\mathbb{Z}}{n\mathbb{Z}})^*$ such that g has order $(p-1)p$ in the subgroup $(\frac{\mathbb{Z}}{p^2q\mathbb{Z}})^*$.
- Let $h = g^n \pmod n$.

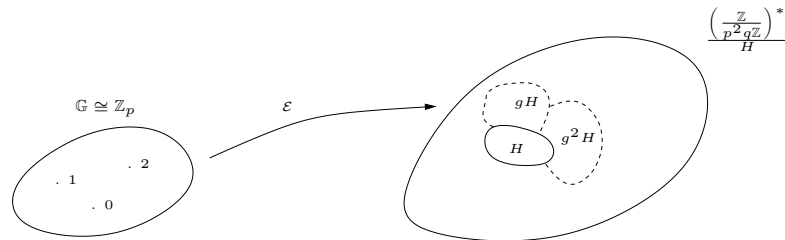


Figure 4: The Okamoto-Uchiyama cryptosystem

Encryption Select $r \in \frac{\mathbb{Z}}{n\mathbb{Z}}$ at random. $m \in \frac{\mathbb{Z}}{p\mathbb{Z}}$;

$$c = g^m h^r \pmod{n}.$$

Decryption Define $L(x) = \frac{x-1}{p}$. Then

$$m = \frac{L(c^{p-1} \pmod{p^2})}{L(g^{p-1} \pmod{p^2})} \pmod{p}.$$

The L operator can be thought of as a logarithm on the group $H = \{x : x \equiv 1 \pmod{p}\}$, and we can show that c^{p-1} and g^{p-1} are in H using Fermat's little theorem:

$$\forall x \vdash p \nmid x : p \mid x^{p-1} - 1.$$

This is the first cryptosystem in this flavour that does not perform some kind of verification process to find the decryption of a ciphertext, but actually computes the plaintext directly.

Correctness of decryption The group $(\mathbb{Z}/n\mathbb{Z})^*$ is isomorphic to $(\mathbb{Z}/p^2\mathbb{Z})^* \times (\mathbb{Z}/q\mathbb{Z})^*$, since $n = p^2q$ with p and q prime numbers.

The group $(\mathbb{Z}/p^2\mathbb{Z})^*$ has a unique subgroup of order p , name as H . By the uniqueness of H , it must be that $H = \{x : x \equiv 1 \pmod{p}\}$.

For any element $x \in (\mathbb{Z}/p^2\mathbb{Z})^*$, we have $x^{p-1} \pmod{p^2} \in H$, since p divides $x^{p-1} - 1$.

The map L is a logarithm from the cyclic group H to the additive group $\mathbb{Z}/p\mathbb{Z}$, $L(ab) = L(a) + L(b)$, and L is an isomorphism between these two groups. It can be thought of as, in the case with the usual logarithm, $L(x)/L(g)$ is the logarithm of x with base g .

We have $(g^m h^r)^{p-1} = (g^m g^{nr})^{p-1} = (g^{p-1})^m g^{p(p-1)rpq} = (g^{p-1})^m \pmod{p^2}$.

To recover m one needs to take the logarithm with base g^{p-1} , which is accomplished by $\frac{L((g^{p-1})^m)}{L(g^{p-1})} = m \pmod{p}$.

2.2.5 Paillier cryptosystem

The Paillier cryptosystem [32] is the famous cryptosystem that we wanted to cover in the first place. As one of the most well-known homomorphic encryption schemes, it is an improvement of the Okamoto-Uchiyama cryptosystem, that decreases the expansion from 3 to 2. This cryptosystem is based on the Composite Residuosity (CR) assumption, meaning that the problem of computing n -th residue classes is computationally difficult.

Paillier came back to $n = pq$, with $\gcd(n, \phi(n)) = 1$, but with the group $\mathbb{G} = \mathbb{Z}_{n^2}^*$, and a proper choice of H led to $k = l(n)$. The encryption cost is reasonably low.

Decryption needs one exponentiation modulo n^2 to the power $\lambda(n)$, and a multiplication modulo n , and it can be managed efficiently through the Chinese Remainder Theorem.

Applying it to Paillier's original scheme, there are several stronger variants proposed. Cramer and Shoup [12] proposed a general approach to gain security against adaptive chosen-ciphertext attacks for certain cryptosystems with some particular algebraic properties. Bresson *et al.* [10] proposed a slightly different version that may be more accurate for some applications.

The main observation for this cryptosystem is the following equation:

$$(1 + x)^m = 1 + mx \pmod{x^2}.$$

Mathematical Basis The security of this system relies on the problem of deciding n -th residuosity.

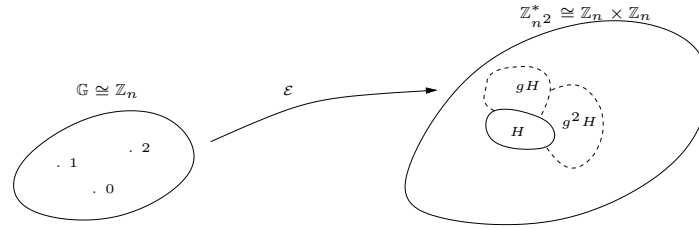


Figure 5: The Paillier cryptosystem

The Cryptosystem

Prerequisites

- Choose two large primes p, q randomly and independently of each other.
- Compute $n = pq$ and $\lambda = \text{lcm}(p-1, q-1)$.
- Select a random $g \in \mathbb{Z}_{n^2}^*$, ensure n divides the order of g , by checking the existence of the following modular multiplicative inverse $\mu = (L(g^\lambda \pmod{n^2}))^{-1} \pmod{n}$.
- The function L is defined as $L(u) = \frac{u-1}{n}$.
- The public key is (n, g) , the private key λ .

Encryption Let m be a plaintext to be encrypted where $m \in \mathbb{Z}_n$

Select random r where $r \in \mathbb{Z}_n^*$.

Compute ciphertext as: $c = g^m \cdot r^n \pmod{n^2}$.

Decryption Ciphertext $c \in \mathbb{Z}_{n^2}^*$

Based upon the $L(x)$ function as defined by Okamoto-uchiyama in Section 2.2.4, compute plaintext: $m = L(c^\lambda \bmod n^2) \cdot \mu \bmod n$

Decryption is essentially one exponentiation modulo n^2 .

The analysis of the main practical aspects of computations required by the cryptosystem and various implementation strategies for increased performance, as well as how the Chinese Remainder Theorem can be used to efficiently reduce the decryption workload of the cryptosystem, are provided by the original paper [32].

Homomorphic properties The scheme is an additive homomorphic cryptosystem; this means that, given only the public-key and the encryption of m_1 and m_2 , one can compute the encryption of $m_1 + m_2$. The homomorphic properties of Paillier's scheme.

$$\begin{aligned} D(\mathcal{E}_{r_1}(m_1)\mathcal{E}_{r_2}(m_2) \bmod n^2) &= m_1 + m_2 \bmod n \\ D(\mathcal{E}_r(m_1)g^{m_2}) &= m_1 + m_2 \bmod n \\ D(\mathcal{E}_r(m_1)^{m_2}) &= m_1 \cdot m_2 \bmod n \end{aligned}$$

The cryptosystem explained above provides semantic security against chosen-plaintext attacks (IND-CPA). The ability to successfully distinguish the challenge ciphertext essentially amounts to the ability to decide composite residuosity, as the decisional composite residuosity assumption (DCRA) is believed to be intractable. Because of the aforementioned homomorphic properties however, the system is malleable, and therefore does not enjoy the highest echelon of semantic security that protects against adaptive chosen-ciphertext attacks (IND-CCA2).

2.2.6 Damgård-Jurik cryptosystem

The Damgård-Jurik cryptosystem [13] is a generalization of the Paillier cryptosystem to groups $\mathbb{Z}_{n^{s+1}}^*$ with $s > 0$. It uses computations modulo m^{s+1} , where n is an RSA modulus and s a natural number. The expansion is $1 + 1/s$, and hence the larger the s is, the smaller the expansion is, which can be close to 1 if s is sufficiently large.

Paillier's cryptosystem is the special case with $s = 1$. The ratio of the encryption cost of this scheme over Paillier's can be estimated to be $(1/6)s(s+1)(s+2)$, while the ratio for the decryption step equals $(1/6)(s+1)(s+2)$. It shows that even if this scheme is better than Paillier's according to its lower expansion, it remains more costly. Moreover, to encrypt or decrypt k blocks of $l(n)$ bits, running Paillier's cryptosystem k times is less costly than running Damgård-Jurik cryptosystem once.

The authors show that the security of Damgård-Jurik cryptosystem is equivalent to that of Paillier, and the security of Damgård-Jurik can be proven under the decisional composite residuosity assumption.

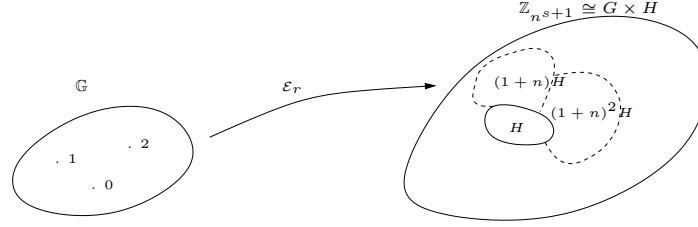


Figure 6: The Damgård-Jurik cryptosystem

The Cryptosystem

Prerequisites $\mathbb{Z}_{n^{s+1}}^* \cong G \times H$, with G a cyclic group of order n^s , and $H \cong \mathbb{Z}_n^*$.

- Choose primes p and q , and compute $n = pq$ and $\lambda = \text{lcm}(p-1, q-1)$.
- Choose $g \in \mathbb{Z}_{n^{s+1}}^*$ such that $g = (1+n)^j x \pmod{n^{s+1}}$, with $\text{gcd}(j, n) = 1$, and $x \in H$.
- Choose d such that $d \pmod{n} \in \mathbb{Z}_n^*$ and $d \equiv 0 \pmod{\lambda}$ (using the Chinese Remainder Theorem). In Paillier's original scheme, $d = \lambda$.
- The public key is (n, g) , the private key d .

Encryption Encryption of a message into the corresponding coset of the factor group G/H :

$$c = g^{m_r n^s} \pmod{n^{s+1}}$$

With $m \in \mathbb{Z}_{n^s}$ the plaintext, and $r \xleftarrow{R} \mathbb{Z}_{n^{s+1}}^*$.

Decryption Given a ciphertext $c \in \mathbb{Z}_{n^{s+1}}^*$, first compute $c^d \pmod{n^{s+1}}$. Clearly, if is a valid ciphertext, we get

$$c^d = (g^{m_r n^s})^d = ((1+n)^{jm} x^{m_r n^s})^d = (1+n)^{jmd \pmod{n^s}} (x^{m_r n^s})^{d \pmod{\lambda}} = (1+n)^{jmd \pmod{n^s}}$$

Apply a recursive version of the decryption mechanism used in Paillier's scheme to obtain jmd . Because of the knowledge of jd , $m = (jmd) \cdot (jd)^{-1} \pmod{n^s}$ can be computed.

Clearly, this system is additively homomorphic over $\mathbb{Z}_{n^s}^*$, that is, the product of encryptions of messages m_1, m_2 is an encryption of $m_1 + m_2 \pmod{n^s}$.

Proofs Now apply the algorithm from the proof of Theorem below one can compute $jmd \bmod n^s$:

Theorem. For any admissible n and $s < p, q$, the map $\psi_s : \mathbb{Z}_{n^s} \times \mathbb{Z}_n^* \mapsto \mathbb{Z}_{n^{s+1}}^*$ given by $(x, r) \mapsto (1+n)^x r^{n^s} \bmod n^{s+1}$ can be inverted in polynomial time given $\lambda(n)$, the least common multiple of $p-1$ and $q-1$.

We refer to the original paper by Damgård-Jurik [13] for the details of the proof.

2.3 Pairing-based extensions

Homomorphic properties as described in Section 2.2 are interesting to have, but they are not that flexible. In fact, they are only useful when one wants to perform only additions in the encrypted domain while it is much more interesting to be able to perform different operations using the same encryption scheme.

An interesting proposal by Boneh *et al.* [8] extends DCRA-based public-key encryption schemes such that apart from additions, one multiplication can be performed in the encrypted domain. This result is achieved by introducing a new type of operation: a pairing.

DEFINITION 4 (bilinearPairing (Bilinear pairing) Let G_1, G_2 be cyclic multiplicative groups of prime order w such that the discrete logarithm problem in both is hard. A bilinear pairing is a map $\hat{e} : G_1 \times G_1 \mapsto G_2$ that satisfies the following properties [6, 7, 9]:

1. **Bilinearity** For all $P, P' \in \mathbb{G}_1$, and all $Q, Q' \in \mathbb{G}_1$ we have

$$e(P + P', Q) = e(P, Q)e(P', Q), \text{ and } e(P, Q + Q') = e(P, Q)e(P, Q').$$

2. **Non-degeneracy**

- For all $P \in \mathbb{G}_1$, with $P \neq 0$, there is some $Q \in \mathbb{G}_2$ such that $e(P, Q) \neq 1$.
- For all $Q \in \mathbb{G}_1$, with $Q \neq 0$, there is some $P \in \mathbb{G}_1$ such that $e(P, Q) \neq 1$.

3. **Computability** The map \hat{e} is efficiently computable.

□

The two main examples of pairings are the Weil and Tate pairing on elliptic curves over finite fields. A detailed exposition of these pairings is not within the scope of this document (yet). However, the advantage of constructing homomorphic cryptosystems based on pairings, is that one can consider the pairings as being black boxes. Details about their construction can be ignored when building the cryptosystems.

Weil Pairings are proposed in [17] as a possible direction for the construction of new homomorphic encryption systems. For further information on pairing-based cryptography, we refer to Galbraith *et al.* [18].

2.3.1 Evaluating 2-DNF formulas on Ciphertexts

We introduce the homomorphic public key encryption scheme that is introduced by Boneh *et al.* [8].

Prerequisites Generate a tuple $(q_1, q_2, \mathbb{G}, \mathbb{G}_1, e)$. Let $n = q_1 q_2$, and pick generators $g, u \xleftarrow{R} \mathbb{G}$, and set $h = u^{q_2}$. The public key is $(n, \mathbb{G}, \mathbb{G}_1, e, g, h)$, the private key q_1 .

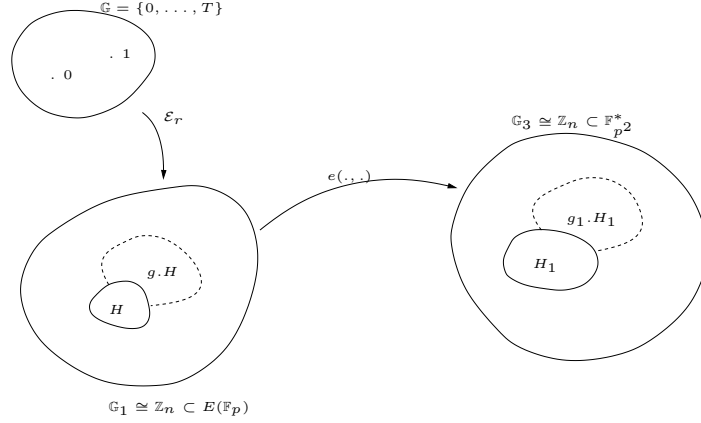


Figure 7: Paired homomorphic groups

Encryption.

$$c = g^m h^r \in \mathbb{G},$$

with $m \in \{0, \dots, T\}; T < q_2$, and $r \xleftarrow{R} \mathbb{Z}_n$.

Decryption. Observe that

$$c^{q_1} = (g^m h^r)^{q_1} = (g^{q_1})^m.$$

To recover m , it suffices to compute the discrete log of c^{q_1} with basis g^{q_1} . Since m is small ($\leq T$), one can deploy the Pollard's lambda method, and compute the discrete log in $O(\sqrt{T})$.

Homomorphic properties. Besides being additively homomorphic, this cryptosystem is also multiplicative (once!). On the ciphertexts of m_1 and m_2 , one can compute the ciphertext of $m_1 + m_2$. Using the bilinear map, $m_1 m_2$ can be computed *once*.

Multiplication of the ciphertexts of m_1 and m_2 can be achieved as follows:

$$\begin{aligned}
c = e(c_1, c_2)h_1^r &= e(g^{m_1}h^{r_1}, g^{m_2}h^{r_2})h_1^r \\
&= g_1^{m_1m_2}h_1^{m_1r_2+m_2r_1+\alpha q_2r_1r_2+r} \\
&= g_1^{m_1m_2}h_1^{r'} \in \mathbb{G}_1
\end{aligned}$$

where $r' = m_1r_2 + m_2r_1 + \alpha q_2r_1r_2 + r$ is distributed uniformly in \mathbb{Z}_n .

Using this cryptosystem, multivariate equations on x_1, \dots, x_i of degree 2 can be computed in the encrypted domain. This enables the construction of SFE (Secure Function Evaluation, see Sect. 3.1) systems for 2-DNF function families.

2.4 ElGamal class of homomorphic encryption schemes

ElGamal cryptosystem [16], proposed by Taher ElGamal in 1985, is a public-key cryptosystem based on the Diffie-Hellman key agreement. ElGamal encryption can be defined over any cyclic group G . The security of the ElGamal scheme depends on the properties of the underlying group G as well as any padding scheme used on the messages. If the computational Diffie-Hellman assumption holds in the underlying cyclic group G , then the encryption function is one-way. If the decisional Diffie-Hellman assumption (DDH) holds in G , then ElGamal achieves semantic security. Semantic security is not implied by the computational Diffie-Hellman assumption alone. The *confidentiality* of ElGamal encryption is equivalent to the CDH assumption, though the *semantic security* of the scheme is based on the decisional Diffie-Hellman assumption.

- Cyclic group G , and a generator g
- Private key s
- Public key $h = g^s$
- $E(h, m) = (g^r, mh^r)$
- $D(s, c, d) = d/(c^s) = mh^r/g^{rs} = m$.

Some nice properties:

- Semantic security (under DDH assumption)
- Ciphertexts can be re-randomized
- Homomorphic multiplication
- Supports precomputation
- Conducive to ZK proofs

Computational Diffie-Hellman (CDH) assumption Computational DH: Given (g, g^a, g^b) , output g^{ab} .

The CDH assumption is related to the *discrete logarithm assumption*, which states that computing the discrete logarithm of a value base a generator g is hard. If taking discrete logs were easy, then the CDH assumption would be false.

Decisional Diffie-Hellman (DDH) assumption Decisional DH: Given (g, g^a, g^b) , distinguish g^{ab} and g^c . I.e., DDH assumption states that it is hard to distinguish tuples of the form (g, g^a, g^b, g^{ab}) from random tuples

For this reason, DDH is considered a stronger assumption than discrete log assumption, in the following sense: there are groups for which detecting DDH tuples is easy, but computing discrete logs is believed to be hard. Thus, requiring that the DDH assumption holds in a group is a more restricting requirement.

Similar to above, DDH assumption is considered a stronger assumption than CDH assumption. If it were possible to efficiently compute g^{ab} from (g^a, g^b) , then one could easily distinguish the two probability distributions above.

If Bob has a non-negligible advantage in winning the IND-CPA game (in the case of ElGamal), we can use him as an oracle for solving the DDH. Hence, if DDH is hard, then ElGamal is semantically secure.

DDH is thought to be hard in several efficiently computable groups, but not in \mathbb{Z}_p^* , where p is prime. This is because given g^a and g^b , one can efficiently compute the Legendre symbol of g^{ab} , giving a successful method to distinguish g^{ab} from a random group element.

ElGamal: Semantically secure, not CCA2 secure, under DDH assumption. Cramer-Shoup: CCA2 secure under DDH assumption.

2.4.1 Cramer-Shoup cryptosystem

The Cramer-Shoup cryptosystem [12], as an extension of the ElGamal cryptosystem, is a general approach to gain security against adaptive chosen-ciphertext attacks for certain cryptosystems with some particular algebraic properties. Its security is based on the computational intractability of the decisional Diffie-Hellman assumption. In contrast to ElGamal, which is extremely malleable, Cramer-Shoup adds additional elements to ensure non-malleability even against a resourceful attacker. This non-malleability is achieved through the use of a collision-resistant hash function and additional computations, resulting in a ciphertext which is twice as large as in ElGamal.

Prerequisites

- Generate an efficient description of a cyclic group G of order q with two distinct, random generators g_1, g_2 .
- Choose five random values $(x_1, x_2, y_1, y_2, z) \in \{0, \dots, q-1\}$, and compute $c = g_1^{x_1} g_2^{x_2}, d = g_1^{y_1} g_2^{y_2}, h = g_1^z$.
- The public key is $(c, d, h, G, q, g_1, g_2)$
- The private key is (x_1, x_2, y_1, y_2, z)

The group can be shared between users of the system.

Encryption To encrypt a message $m \in G$ under a public key as above, one does the following.

- Convert plaintext m into an element of G .
- Choose a random $k \in \{0, \dots, q-1\}$, then calculates:

$$u_1 = g_1^k, u_2 = g_2^k$$

$$e = h^k m$$

$$\alpha = H(u_1, u_2, e),$$

where $H()$ is a collision-resistant hash function.

$$v = c^k d^{k\alpha}$$

- The ciphertext is (u_1, u_2, e, v) to Alice.

Decryption To decrypt a ciphertext (u_1, u_2, e, v) under a secret key as above one does the following.

- Compute $\alpha = H(u_1, u_2, e)$ and verifies that $u_1^{x_1} u_2^{x_2} (u_1^{y_1} u_2^{y_2})^\alpha = v$. If this test fails, further decryption is aborted and the output is rejected.
- Otherwise, the plaintext is $m = e / (u_1^z)$.

The decryption stage correctly decrypts any properly-formed ciphertext, since $u_1^z = g_1^{kz} = h^k$ and $m = e / h^k$.

If the space of possible messages is larger than the size of G , then the message can be split into several pieces and each piece can be encrypted independently. Alternately, Cramer-Shoup cryptosystem may be used in a hybrid cryptosystem to improve efficiency on long messages.

2.4.2 ElGamal-Paillier amalgam

The ElGamal-Paillier amalgam, proposed by Damgård and Jurik in 2003 [14], merges Paillier and the additively homomorphic variant of ElGamal, based on DDH and RSA-like assumptions. More precisely, it is based on Damgård-Jurik cryptosystem and Cramer-Shoup cryptosystem. The goal was to gain the advantages of both schemes while minimizing their drawbacks.

Key generation

- Choose an RSA modulus $n = pq$ of length k bits, with $p = 2p' + 1$ and $q = 2q' + 1$ where p, q, p', q' are primes.
- Select an element $g \in Q_n$, the group of all squares of \mathbb{Z}_n^* , and $\alpha \in \mathbb{Z}_\tau$ where $\tau = p'q' = |Q_n|$.
- The public key is then (n, g, h) with $h = g^\alpha \pmod n$ and the private key is α .

Encryption Given a plaintext $m \in \mathbb{Z}^+$, choose an integer $s > 0$ such that $m \in \mathbb{Z}_{n^s}$ and a random $r \in \mathbb{Z}_N$, and the ciphertext is:

$$E_s(m, r) = (g^r \pmod n, (h^r \pmod n)_{n^s} (n+1)^m \pmod{n^{s+1}})$$

Decryption Given a ciphertext $c = (G, H) = E_s(m, r)$, s can be deduced from the length of c (or attached to the encryption) and the plaintext m can be found as:

$$\begin{aligned} m &= L_s(H(G^\alpha \pmod n)^{-n^s}) \\ &= L_s((g^{\alpha r} \pmod n)^{n^s} (n+1)^m (g^{\alpha r} \pmod n)^{-n^s}) \\ &= L_s((n+1)^m \pmod{n^{s+1}}) = m \pmod{n^s} \end{aligned}$$

Note that the key generation above assumes that one knows τ and hence the factorization when choosing α . However, one can also choose $\alpha \in_R \mathbb{Z}_N$, that is, generate $\alpha, h = g^\alpha$ from n, g only. This makes no difference to security, since h will have an indistinguishable distribution, and it allows to have n, g be system constants used by all users.

Pros and cons

Pros:

- Full decryption of message m ;

- Expensive Distributed Key Generation is only at system setup, with a single, system-wide RSA modulus n for all users.

Cons:

- A large overhead due to large ciphertexts, e.g. compared to ElGamal combined with elliptic curves, even if secure computation is mostly bitwise with Boolean circuits.
- Security is based on two assumptions: the factorization of RSA modulus n is actually a trapdoor and may get compromised.

2.5 Lattice-based full homomorphic encryption

A third class of homomorphic encryption schemes that we wish to introduce originates from research in lattice-based cryptography. This is a rather new area in cryptography.

An integer lattice \mathcal{L} is a n -dimensional module of the vector space \mathbb{Z}^n . The basis of a lattice is a set of n linear independent vectors. As usual with modules and vector spaces, a lattice can have many different bases.

There are two hardness assumptions that have been identified in lattices that are of interest for cryptographic schemes.

Shortest Vector Problem (SVP). Given a basis of a lattice, it seems to be untractable to find the shortest vector of the lattice.

Closest Vector Problem (CVP). Given a basis of a lattice and a vector in \mathbb{Z}^n (which is not in that lattice), it is hard to find the vector in \mathcal{L} that is the closest to that vector.

The big advantage of lattice-based cryptography is the fact that it seems to be that the hardness assumptions are much stronger than in traditional cryptography. Even when quantum computers would become a reality and solve the factorization problem (and hence defeat many hardness assumptions including the DCRA), it seems that these will not work against lattice-based schemes. However, we need to stress that research in lattice-based cryptography is still in its very early steps.

An example of a public key encryption scheme that is based on a lattice hardness assumption is the cryptosystem proposed in 2005 by Regev [37] and is based on the *Learning With Errors* (LWE) problem. In this system, the private key is a vector s chosen at random from \mathbb{Z}_q^n . The public key is the inner product of a set of vectors a_1, \dots, a_m chosen independently and uniform at random from \mathbb{Z}_q^n , with the private vector and includes an error offset e_1, \dots, e_m :

$$\text{PK} = (a_i, b_i = \langle a_i, s \rangle / q + e_i)_{i=1}^m.$$

The main idea is that without the knowledge of s , no adversary can remove the error offset, and hence decrypt correctly. This relates to the Closest Vector Problem. We refer to [37] for the details of this encryption scheme.

A fully homomorphic encryption scheme. Recently, Craig Gentry [19] introduced a first fully homomorphic encryption scheme. This encryption scheme is also based on the LWE hardness assumption, where an error offset disturbs the decryption of ciphertext. In this scheme, both addition and multiplication are feasible in the encrypted domain. Addition is naturally feasible on lattices, since two points (vectors) can be added with conventional algebraic formulas. The multiplication in the encrypted domain consists of operations on ideal lattices.

Although this might sound simple, it is definitely not. The tricky thing is that the error offset will increase when operations are performed in the encrypted domain. After a few operations, the error offset might be so large that the decryption will fail, since another vector in the lattice might be closer than the intended vector. Note that the encryption of a plaintext is not a vector in the original lattice, due to the error offset, but a vector that is *close* to a vector – due to the CVP it is assumed to be intractable to decide which one.

The main result by Gentry [19] is the bootstrapping process to make sure that the decryption of a ciphertext that has been obtained after several operations in the encrypted domain, is still correct. The idea behind this, is that after each operation, a special operation is included that reduces the error offset without revealing any private information. In other words, after an operation on vectors *close* to lattice vectors has been performed, the resulting vector needs to be brought *closer* to the intended vector, without revealing the *closest* vector.

We refer to the original publication by Gentry [19] for the details of this scheme. Given that this result has been presented only very recently, we were not yet able to scrutinize this result.

Nevertheless, the impact of this result cannot be underestimated! A fully homomorphic public key encryption scheme is a holy grail in cryptography and mathematics and seemed to be a utopia until recently. The existence of such a scheme was doubtful, but its purposes endless. For example, a *practical* homomorphic encryption scheme allows the construction of circuits that are able to perform a wide range of functions in the encrypted domain.

Exciting times lay ahead in this area of cryptography and mathematics.

3 Computing with Encrypted Functions

In 1982, Yao [41] presented a solution for the millionaires problem, where two millionaires want to know whom is the richest, without revealing their proper wealth. This led to the concept of *Secure Function Evaluation (SFE)* with seminal work of Goldreich, Micali, and Wigderson [20]. Although this initially started as research on preserving the confidentiality of data, in this section we will show how this can lead to new insights for computing with encrypted functions.

3.1 Secure Function Evaluation

The concept of Secure Function Evaluation was introduced within the context of two-party computation. Denote with Alice (A) and Bob (B) the parties that are able to communicate with each other. Both parties wish to compute a (known) function f on their respective inputs x_A and x_B , while keeping their input private to the other parties. There is no third party that can accept their inputs; they have to compute the result using a protocol between each other. A two-party computation protocol is denoted secure if no party can learn more from the input of the other party than he could learn given the public function, the result and his own input. Extensions towards multi-party computation have been developed, but are beyond the scope of the research that is conducted in this task.

Garbled Circuits. The main idea in techniques for secure function evaluation, is the use of *garbled circuits*. These are circuits where the values of the wires are *randomly* assigned to 0 or 1 (garbling process), while the lookup tables that represent the gate computations are re-computed accordingly (garbled tables).

The two-party computation protocol proceeds as follows: Alice computes a garbled circuit that corresponds to the circuit of the public function f , and sends this garbled circuit to Bob. Bob is able to evaluate this circuit with his private input to compute the result. The process of evaluation will require Bob to communicate with Alice such that the garbled gates can be evaluated with respect to her private input. The idea here is that Bob should not be able to know the value of Alice's input (which is obtained due to the fact that the circuit is Garbled in a random way by Alice), while Alice should not be able to know which entries of the gate lookup table were of interest to Bob (and hence deduce information on Bob's input). The latter is achieved by *Oblivious Transfer (OT)* [36].

A practical construction was presented by Goldreich *et al.* [20], which applies to circuits that consist of \wedge (AND) and \vee (OR) operations. The \vee gates can be computed via a homomorphic operation, while the evaluation of the \wedge gates occurs via an Oblivious Transfer protocol between Alice and Bob. This, and subsequent constructions were designed within the context of an *honest-but-curious adversary*. This basically means that the adversary (one of the two parties) will attempt to find information on the input of the other party, but will behave according to the protocol.

There are three main strategies in which a party could deviate from the protocol in order to defeat the objective of two-party computations:

- Alice might cheat in the construction of the garbled circuit. To enforce honest behavior by Alice, the *cut-and-choose* technique has been introduced. The idea is that Alice presents several garbled circuits of the function f , after which Bob will ask Alice to reveal a number of them (randomly selected by Bob) in order to proof their correctness. After that, a high probability of confidence in the construction of the garbled circuits is established, and Bob may assume that the remaining un-revealed circuits are correct. Subsequent work includes further efficiency improvements [30].
- Bob might deviate in the oblivious transfer protocol, in order to obtain both lookup table entries that correspond to Alice's input, and as a result compute extra information on her input. Zero knowledge protocols have been introduced to establish a degree of confidence that Bob behaves correctly.
- At the end of the protocol, Alice will know the result of the evaluation. She might refuse to present this result to Bob.

Solutions to address the above issues, and results to reduce the number of communication rounds between the parties have been conducted in subsequent work [38, 26, 28, 34, 27]. We refer to the work by Herzberg *et al.* [24] for more details and subsequent work that was conducted within the context of this project.

3.2 Universal Circuits

In [39], Valiant showed that there exists a combinatorial (acyclic) Boolean circuit of complexity $O(s \log s)$ that can be made to compute any Boolean function of complexity s by setting a specially designed set of control inputs to appropriate fixed values. Such *universal circuits* can be used to bridge between computing on encrypted data and computing with encrypted functions.

The evaluation of a function f on the input x can be performed by a Universal Circuit, where the control inputs are set to c_f , and the other inputs to values i_x that correspond to x . As a result:

$$f(x) = \text{UC}(c_f, i_x),$$

where UC is a Universal Circuit that consists of basic building blocks such as \wedge and \neg gates. However, when we wish to deploy this strategy in practice, this approach will lead to unreasonably large implementations even for small operations. Moreover, a direct deployment of this approach does not offer indistinguishability between executions. Hence, it remains to be seen up to what extent this approach is valuable, and how they can be improved for efficiency and security (indistinguishability). This has not been within the scope of this project.

4 Towards RE-TRUST solutions based on CED/CEF

In this section, we describe how techniques for Computing on Encrypted Data and Computed with Encrypted Functions can be beneficial for the development of practical solutions for the RE-TRUST project.

4.1 White-Box Remote Program Execution

The objective of *White-Box Remote Program Execution* (WBRPE) is to address the issue of secure execution of software on remote, untrusted computing platforms for generic programs.

Obfuscation. One common approach to protect remotely executing code, is by sending an *obfuscated* version of the code to the remote execution platform. An obfuscated version $\mathcal{O}(P)$ of a program P is a program that is functionally equivalent to its original (that is, $\forall x : \mathcal{O}(P)(x) = P(x)$), which does not leak any useful information. The security is captured by the virtual black-box property, which compares the information obtained by analysis of the obfuscated code to the information that can be obtained by having only black-box (functional) access to the program. An obfuscator is denoted to be secure, when no additional information can be extracted. This captures that an obfuscated program is hard to analyze and reverse engineer. Unfortunately, Barak *et al.* [1] have shown that it is impossible to construct an obfuscator \mathcal{O} that meets this security requirement for all programs or even natural families of programs (such as encryption schemes or digital signature schemes).

Towards a practical solution. The WBRPE approach takes a different strategy in the sense that protected applications do not need to be functionally equivalent. Instead, an encrypted input is provided to the code on the untrusted machine, which produces encrypted output that the originator is able to parse (decrypt).

Figure 8 depicts the WBRPE architecture. The main component in the architecture is the *Obfuscated Virtual Machine* (OVM), which takes as input an encrypted string $\varepsilon(P, i)$ produced by trusted entity (which includes a program description P and server input i), and input a that originates from the untrusted execution platform. The output it produces is the garbled string $\varepsilon'(P(i, a))$.

In [papers by herzberg], details on this architecture are described. [say something about robust combiners]. Practical solutions [Abstract for RE-TRUST 2009 reference] are being developed, which essentially are based on computations performed by garbled circuits (See Sect. 3.1). All these publications have been developed within the context of the RE-TRUST project.

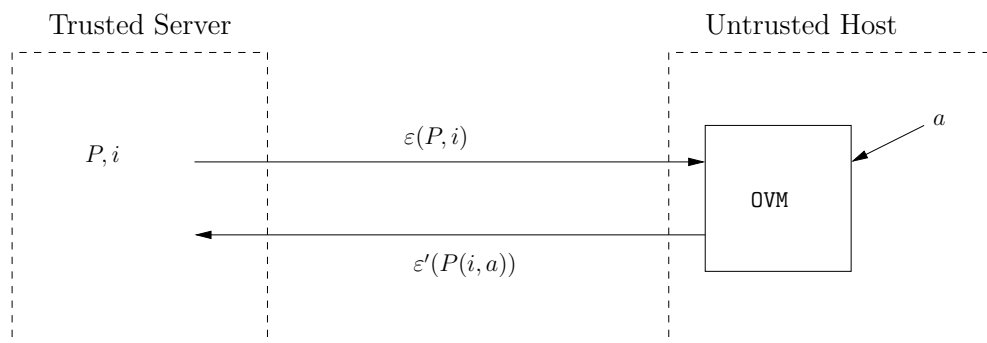


Figure 8: White-Box Remote Program Execution Architecture

References

- [1] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (Im)possibility of Obfuscating Programs. In *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18. Springer-Verlag, 2001.
- [2] Cataldo Basile, Yoram Ofek, Alessandro Zorat, Mariano Ceccato, Paolo Tonella, Jan Cappaert, Dries Schellekens, Brecht Wyseur, Jerome D’Annoville, Igor Kottenko, Vasily Desnitsky, Paolo Falcarin, Stefano Di Carlo, Amir Herzberg, and Haya Shulman. Security Analysis. RE-TRUST deliverable 4.X, 2009.
- [3] Mihir Bellare, Anand Desai, E. Joriki, and Phillip Rogaway. A Concrete Security Treatment of Symmetric Encryption. In *Proceedings of the 38th Symposium on Foundations of Computer Science (FOCS 1997)*, IEEE Computer Society, pages 394–403, 1997.
- [4] Mihir Bellare and Phillip Rogaway. Optimal Asymmetric Encryption - How to Encrypt with RSA. In *Advances in Cryptology - EUROCRYPT 1994*, volume 950 of *Lecture Notes in Computer Science*, pages 341–358. Springer, 1994.
- [5] Josh Benaloh. Dense probabilistic encryption. pages 120–128.
- [6] Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. In *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer-Verlag, 2001.
- [7] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer-Verlag, 2003.
- [8] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In Joe Killian, editor, *Proceedings of Theory of Cryptography Conference 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–342. Springer, 2005.
- [9] Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signatures from the Weil Pairing. In *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532, London, UK, 2001. Springer-Verlag.
- [10] Emmanuel Bresson, Dario Catalano, and David Pointcheval. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In *Advances in Cryptology - ASIACRYPT 2003*, Lecture Notes in Computer Science. Springer-Verlag.
- [11] Stefano Di Carlo, Jasvir Nagra, and Brecht Wyseur. Software-based method – initial architecture. RE-TRUST deliverable 1.2, 2007.

- [12] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. pages 45–64. Springer-Verlag, 2001.
- [13] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier’s probabilistic public-key system. In *In proceedings of PKC 01*, Lecture Notes in Computer Science, pages 119–136. Springer-Verlag, 2001.
- [14] Ivan B. Damgård and Mads J. Jurik. A length-flexible threshold cryptosystem with applications. In *In proceedings of ACISP 03*, Lecture Notes in Computer Science, pages 350–364, 2003.
- [15] Josep Domingo-Ferrer and Vicenç Torra. A critique of k-anonymity and some of its enhancements. In *ARES*, pages 990–993. IEEE Computer Society, 2008.
- [16] Taher ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, 1985.
- [17] Caroline Fontaine and Fabien Galand. A survey of homomorphic encryption for non-specialists. 2007.
- [18] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Appl. Math.*, 156(16):3113–3121, 2008.
- [19] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *STOC*, pages 169–178. ACM, 2009.
- [20] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *Proceedings of the 19th ACM Symposium on Theory of Computing (STOC 1987)*, pages 218–229. ACM Press, 1987.
- [21] Shafi Goldwasser and Silvio Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *STOC ’82: Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 365–377, New York, NY, USA, 1982. ACM.
- [22] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information. In *Proceedings of the 14th ACM Symposium on Theory of Computing (STOC 1982)*, pages 365–377. ACM Press, 1982.
- [23] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
- [24] Amir Herzberg and Haya Shulman. Practical Secure Remote Computing. RE-TRUST 2009 workshop, work in progress, 2009.

- [25] Marc Joye, Jean jacques Quisquater, and Moti Yung. On the power of misbehaving adversaries and security analysis of the original epoc. In *In CT – RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 208–222. Springer-Verlag, 2001.
- [26] Vladimir Kolesnikov. Gate evaluation secret sharing and secure one-round twoparty computation. In *Advances in Cryptology - ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 136–155. Springer-Verlag, 2005.
- [27] Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. Cryptology ePrint Archive, Report 2009/411, 2009. <http://eprint.iacr.org/>.
- [28] Yehuda Lindell and Benny Pinkas. An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries. In *Advances in Cryptology - EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 52–78. Springer-Verlag, 2007.
- [29] David Naccache and Jacques Stern. A new public key cryptosystem based on higher residues. In *ACM Conference on Computer and Communications Security*, pages 59–66, 1998.
- [30] Jesper Buus Nielsen and Claudio Orlandi. Lego for two-party secure computation. In *TCC '09: Proceedings of the 6th Theory of Cryptography Conference on Theory of Cryptography*, pages 368–386, Berlin, Heidelberg, 2009. Springer-Verlag.
- [31] T. Okamoto, S. Uchiyama, and E. Fujisaki. Epoc: efficient probabilistic publickey encryption. Technical report, Contribution to IEEE - describes EPOC-3, 2000.
- [32] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. *Lecture Notes in Computer Science*, 1592:223–??, 1999.
- [33] Pascal Paillier. Impossibility proofs for rsa signatures in the standard model. In Masayuki Abe, editor, *CT-RSA*, volume 4377 of *Lecture Notes in Computer Science*, pages 31–48. Springer, 2007.
- [34] Annika Paus, Ahmad-Reza Sadeghi, and Thomas Schneider. Practical secure evaluation of semi-private functions. Cryptology ePrint Archive, Report 2009/124, 2009. <http://eprint.iacr.org/>.
- [35] S. C. Pohlig and M. E. Hellman. An improved algorithm for computing logarithms over $gf(p)$ and its cryptographic significance. *IEEE Transactions on Information Theory*, IT-24-1:106–110, 1978.
- [36] Michael O. Rabin. How to Exchange Secrets by Oblivious Transfer. Harvard Center for Research in Computer Technology, manuscript (1981).
- [37] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 84–93, New York, NY, USA, 2005. ACM.

- [38] Tomas Sander, Adam Young, and Moti Yung. Non-Interactive CryptoComputing For NC^1 . In *Proceedings of the 40th Symposium on Foundations of Computer Science (FOCS 1999)*, IEEE Computer Society, pages 554–567, 1999.
- [39] Leslie G. Valiant. Universal Circuits (Preliminary Report). In *Proceedings of the 8th ACM Symposium on Theory of Computing (STOC 1976)*, pages 196–203. ACM Press, 1976.
- [40] Brecht Wyseur, Thomas Herlea, Dries Schellekens, and Jasvir Nagra. Hardware/software-based method – initial architecture. RE-TRUST deliverable 1.3, 2007.
- [41] Andrew Chi-Chih Yao. Protocols for Secure Computations (Extended Abstract). In *Proceedings of the 23rd Symposium on Foundations of Computer Science (FOCS 1982)*, IEEE Computer Society, pages 160–164, 1982.