Fast Short and Fast Linear Cramer-Shoup

Pascal Lafourcade[1], Léo Robert[1] and Demba Sow[2]

[1] University Clermont Auvergne, LIMOS, CNRS UMR 6158, Aubière, France
`firstname.lastname@uca.fr`,
[2] LACGAA, University Cheikh Anta Diop of Dakar, Senegal
`demba1.sow@ucad.edu.sn`

Abstract. A linear Cramer-Shoup encryption scheme version was proposed by Shacham in 2007. Short Cramer-Shoup encryption scheme was designed by Abdalla et al. in 2014. This scheme is a variant of the Cramer-Shoup encryption scheme that has a smaller size. They proved that it is an IND-PCA secure encryption under DDH and the collision-resistance assumptions. We design a faster version of Short Cramer-Shoup encryption scheme denoted *Fast Short Cramer-Shoup* encryption. We also, proposed a faster version of linear Cramer-Shoup encryption called *Fast Linear Cramer-Shoup*. We prove that the Fast Short Cramer-Shoup is IND-PCA secure under DDH and the collision-resistance assumptions. We also, show that our linear encryption is CCA secure under the Linear assumption. Finally we run an evaluation of performances of our schemes.

keywords: Short Cramer-Shoup, Linear Cramer-Shoup, Linear Assumption, IND-CCA, IND-PCA.

1 Introduction

Cramer-Shoup cryptosystem was introduced in 1998 by Cramer et al. in [?]. It is an encryption scheme based on ElGamal encryption that is IND-CCA secure. In [?], a linear version of Cramer-Shoup scheme was proposed. A short Cramer-Shoup scheme was also proposed in [?]. This scheme improves the performance of Cramer-Shoup scheme by reducing the number of generators in G and the number of parameters of the keys. This scheme is also IND-PCA secure which is lower security notion than IND-CCA and stronger than IND-CPA. But applied to small messages, IND-PCA implies IND-CCA.

Contributions. Our main aim is to improve the efficiency of Short and Linear Cramer-Shoup public key schemes. Our contributions are as follows:

– We design a cryptographic encryption scheme, called *Fast Short Cramer-Shoup*, based on the Generalized ElGamal encryption scheme [?]. We follow the spirit of Short Cramer-Shoup versions introduced in [?]. We modify the key generation and the decryption algorithm to be faster. We prove its security against Plaintext-Checking Attack (IND-PCA) under the Decisional Diffie-Hellman (DDH) and the collision-resistance assumptions.

– We also design a *Fast Linear Short Cramer-Shoup* scheme. We prove that our linear scheme is secure in the CCA sense if $\mathcal{HF}$ a secure **UOWHF** family and the Linear assumption hold in a group G.
– Finally, we implement all these schemes with GMP [?] to demonstrate that Fast Short Cramer-Shoup and Fast Linear Cramer-Shoup are significantly faster than Short Cramer-Shoup and Linear Cramer-Shoup respectively.

Related works. ElGamal cryptosystem was proposed in 1984 by T. ElGamal in [?]. It was one of the first cryptosystems whose security was based on the problem of the discrete logarithm (DLP). ElGamal's scheme is IND-CPA secure under the Decisional Diffie-Hellman (DDH) hypothesis. Cramer-Shoup is a cryptosystem proposed by Cramer et al. in [?]. It is based on ElGamal's scheme and it is IND-CCA2 secure under the DDH assumption. Many versions based on Original Cramer-Shoup scheme [?] have been introduced. The original Cramer-Shoup's scheme presented in the eprint version [?], then the standard Cramer-Shoup's version published in CRYPTO'98 [?], the efficient Cramer-Shoup's version also proposed in [?] and finally Short Cramer-Shoup's version proposed in [?]. The main difference of Original and Standard Cramer-Shoup schemes is that the Standard scheme uses only one exponent z to compute the public parameter h instead of two exponents z_1 and z_2 in the Original scheme. In Section 4 of [?], the efficient variant of the Cramer-Shoup scheme is presented. Note that Original and Efficient Cramer-Shoup encryption algorithms are exactly the same. But theirs key generation and decryption algorithms are slightly different. In Efficient Cramer-Shoup, the key generation uses less elements and then less exponentiations. Short Cramer-Shoup scheme [?] is a variant of the above Cramer-Shoup scheme [?]. In Short Cramer-Shoup, key generation algorithm uses less generator and less elements in public and secret keys. Original, Standard and Efficient Cramer-Shoup schemes are IND-CCA secure under DDH but Short Cramer-Shoup scheme is IND-PCA secure under the DDH and the collision-resistance assumptions. In [?], Boneh et al. introduced the Decisional Linear Assumption (DLin) and proposed a linear scheme based on ElGamal. The linear ElGamal scheme is IND-CPA secure under the (DLin). In [?], Linear Cramer-Shoup scheme is presented. As Original Cramer-Shoup scheme, the Linear Cramer-Shoup scheme is IND-CCA secure under DDH. We improve both Short and Linear Cramer-Shoup schemes.

Outline. In Section 2, we recall public key encryption and the existing Cramer-Shoup schemes. In Section 3, we present our Fast Short Cramer-Shoup encryption scheme. In Section 4, we also propose a Fast Linear version of Cramer-Shoup. In Section 5, we show the results of our performance evaluations. The security proofs of our proposed schemes are available in [?].

2 Preliminaries

Boneh et al. [?] introduced a Decisional assumption, called Linear, intended to take the place of DDH in groups - in particular, bilinear groups [?] - where DDH is easy. For this setting, the Linear problem has desirable properties, as they have shown: it is hard if DDH is hard, but, at least in generic groups [**?**], it remains hard even if DDH is easy. Let G be a cyclic multiplicative group of prime order p, and let g_1, g_2, and g_3 be arbitrary generators of G, we consider the following problem:

Linear Problem in G: Given $g_1, g_2, g_3, g_1^a, g_2^b, g_3^c \in G$ as input, output yes if $a + b = c$ and no otherwise. The advantage of an algorithm $\mathcal{A}$ in deciding the Linear problem in G is denoted by $\mathbf{Adv}_{\mathcal{A}}^{\text{linear}}$ and it is equal to:

$$|Pr[\mathcal{A}(g_1, g_2, g_3, g_1^a, g_2^b, g_3^{a+b}) = yes : g_1, g_2, g_3 \overset{\$}{\leftarrow} G, a, b \overset{\$}{\leftarrow} \mathbb{Z}_p]$$
$$-Pr[\mathcal{A}(g_1, g_2, g_3, g_1^a, g_2^b, \eta) = yes : g_1, g_2, g_3, \eta \overset{\$}{\leftarrow} G, a, b \overset{\$}{\leftarrow} \mathbb{Z}_p]|$$

with the probability taken over the uniform random choice of the parameters to $\mathcal{A}$ and over the coin tosses of $\mathcal{A}$. We say that an algorithm $\mathcal{A}(t, \epsilon)$-decides Linear in G if $\mathcal{A}$ runs in time at most t, and $\mathbf{Adv}_{\mathcal{A}}^{\text{linear}}$ is at least ϵ.

Definition 1. *We say that the (t, ϵ)-Decision Linear Assumption holds in G if no algorithm (t, ϵ)-decides the Decision Linear problem in G.*

The Linear problem is well defined in any group where DDH is well defined. It is mainly used in bilinear groups like in [**?**,**?**,**?**].

2.1 Original Cramer-Shoup Scheme

We recall the original Cramer-Shoup encryption scheme presented in the eprint version [**?**]. It is composed of a key generation algorithm, an encryption and a decryption algorithm. The decryption algorithm consists into two algorithms one for recovering from the ciphertext the plaintext and one to check the non-malleability of the ciphertext in order to ensure IND-CCA2 security. We define three functions: the setup function, denoted CS.KG(), the encryption function, denoted CS.Enc() and the decryption function, denoted CS.Dec().

CS.KG(1^λ): Select a group G of prime order q. Choose eight random elements: $g_1, g_2 \in G$ and $x_1, x_2, y_1, y_2, z_1, z_2 \in \mathbb{Z}_q$.
Compute in G: $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$ and $h = g_1^{z_1} g_2^{z_2}$. Choose a hash function H that hashes messages to elements of $\mathbb{Z}_q$. Return (pk, sk) where $pk = (g_1, g_2, c, d, h, H)$ and $sk = (x_1, x_2, y_1, y_2, z_1, z_2)$.

CS.Enc(pk, M): To encrypt message m with $pk = (g_1, g_2, c, d, h, H)$, choose a random element $r \in \mathbb{Z}_q$. Compute $u_1 = g_1^r, u_2 = g_2^r, e = h^r m, \alpha = H(u_1, u_2, e)$ and $v = c^r d^{r\alpha}$. Return the following ciphertext: (u_1, u_2, e, v).

CS.Dec(sk, ct): Knowing sk, decrypt a ciphertext (u_1, u_2, e, v). Compute $\alpha = H(u_1, u_2, e)$. Verify if $u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha} = v$. Output $m = e u_1^{-z_1} u_2^{-z_2}$ if the condition holds, otherwise output "reject".

Correctness:

Verification: Since $u_1 = g_1^r$ and $u_2 = g_2^r$, we have: $u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha} = u_1^{x_1} u_2^{x_2}$ $(u_1^{y_1} u_2^{y_2})^\alpha = (g_1^{x_1} g_2^{x_2})^r (g_1^{y_1} g_2^{y_2})^{r\alpha} = c^r d^{r\alpha} = v$.

Decryption: $e u_1^{-z_1} u_2^{-z_2} = h^r m g_1^{-r z_1} g_2^{-r z_2} = h^r m h^{-r} = m$.

2.2 Linear Cramer-Shoup Scheme

We recall the Linear Cramer-Shoup Encryption [?]. We define three functions: the setup function, denoted LCS.KG(), the encryption function, denoted LCS.Enc() and the decryption function, denoted LCS.Dec().

LCS.KG(1^λ)**:** Choose random generators $g_1, g_2, g_3 \overset{\$}{\leftarrow} G$ and exponents $x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3 \overset{\$}{\leftarrow} \mathbb{Z}_p$ and set $c_1 \leftarrow g_1^{x_1} g_3^{x_3}$, $d_1 \leftarrow g_1^{y_1} g_3^{y_3}$, $h_1 \leftarrow g_1^{z_1} g_3^{z_3}$ $c_2 \leftarrow g_2^{x_2} g_3^{x_3}$, $d_2 \leftarrow g_2^{y_2} g_3^{y_3}$, $h_2 \leftarrow g_2^{z_2} g_3^{z_3}$. Choose a **UOWHF** $H \overset{\$}{\leftarrow} \mathcal{HF}$. The public key is $pk = (g_1, g_2, g_3, c_1, c_2, d_1, d_2, h_1, h_2)$; the secret key is $sk = (x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3)$.

LCS.Enc(pk, M)**:** To encrypt a message $M \in G$, using $pk = (g_1, g_2, g_3, c_1, c_2, d_1, d_2, h_1, h_2)$. Choose random exponents $r_1, r_2 \overset{\$}{\leftarrow} \mathbb{Z}_p$, and set $u_1 \leftarrow g_1^{r_1}$, $u_2 \leftarrow g_2^{r_2}$, $u_3 \leftarrow g_3^{r_1+r_2}$ and $e \leftarrow M h_1^{r_1} h_2^{r_2}$; now compute $\alpha \leftarrow H(u_1, u_2, u_3, e)$ and finally, $v \leftarrow (c_1 d_1^{\alpha})^{r_1} (c_2 d_2^{\alpha})^{r_2}$. The ciphertext is $ct = (u_1, u_2, u_3, e, v)$.

LCS.Dec(sk, ct)**:** Parse pk as $(g_1, g_2, g_3, c_1, c_2, d_1, d_2, h_1, h_2, H)$, the private key sk as $(x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3)$ and the ciphertext ct as (u_1, u_2, u_3, e, v). Compute $\alpha \leftarrow H(u_1, u_2, u_3, e)$ and test that $u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2} u_3^{x_3+\alpha y_3} \overset{?}{=} v$ holds. If it does not, output "**reject**".
Otherwise, compute and output $M \leftarrow e/(u_1^{z_1} u_2^{z_2} u_3^{z_3})$.

Correctness: If the keys and encryption are generated according to the algorithms above, the test in **LCS.Dec** is satisfied, since we have

$$\begin{aligned} v &= (c_1 d_1^{\alpha})^{r_1} (c_2 d_2^{\alpha})^{r_2} \\ &= (g_1^{x_1+\alpha y_1} g_3^{x_3+\alpha y_3})^{r_1} \cdot (g_2^{x_2+\alpha y_2} g_3^{x_3+\alpha y_3})^{r_2} \\ &= (g_1^{r_1})^{x_1+\alpha y_1} \cdot (g_2^{r_2})^{x_2+\alpha y_2} \cdot (g_3^{r_1+r_2})^{x_3+\alpha y_3} \\ &= u_1^{x_1+\alpha y_1} \cdot u_2^{x_2+\alpha y_2} \cdot u_3^{x_3+\alpha y_3} \end{aligned}$$

Next, decryption algorithm computes M as follows,

$$\begin{aligned} e/(u_1^{z_1} u_2^{z_2} u_3^{z_3}) &= e/(g_1^{r_1 z_1} g_2^{r_2 z_2} g_3^{(r_1+r_2) z_3}) \\ &= (e)/((g_1^{z_1} g_3^{z_3})^{r_1} (g_2^{z_2} g_3^{z_3})^{r_2}) \\ &= (M \cdot h_1^{r_1} h_2^{r_2}) \cdot (h_1^{r_1} h_2^{r_2}) \\ &= M. \end{aligned}$$

Security proof of Linear Cramer-Shoup (LCS).

Theorem 1. [?]. *LCS scheme is IND-CCA secure if $\mathcal{HF}$ is a secure **UOWHF** family and if the Linear assumption holds in G.*

2.3 Short Cramer-Shoup Scheme

The Short Cramer-Shoup (SCS) encryption scheme [?] is a variant of the above Cramer-Shoup encryption scheme [?], but with one less element. It is defined as follows, in a cyclic group G of prime order p, with a generator g, together with a hash function H randomly drawn from a collision-resistant hash function family $\mathcal{HF}$ [?] from the set $\{0,1\}^* \times G^2$ to the set $G \setminus \{1\}$. We define three functions: the setup function, denoted SCS.KG(), the encryption function, denoted SCS.Enc() and the decryption function, denoted SCS.Dec(). We now describe how these functions work.

SCS.KG(1^λ)**:** Pick five random elements $s, a, b, a', b' \in \mathbb{Z}_p$.
Compute $h = g^s, c = g^a h^b, d = g^{a'} h^{b'}$.
Return (pk, sk), where $pk = (g, h, c, d, H)$ and $sk = (s, a, b, a', b')$.

SCS.Enc(pk, m)**:** To encrypt a message m with $pk = (g, h, c, d, H)$, choose random element $r \in \mathbb{Z}_p$. Compute $u = g^r, e = h^r m, \alpha = H(u, e)$ and $v = (c(d^\alpha))^r$. Output the ciphertext (u, e, v).

SCS.Dec$^\ell(sk, ct)$**:** To decrypt a ciphertext (u, e, v) using sk, compute $\alpha = H(u, e)$. Then compute $m = eu^{-s}$ and check $v = u^{a+a'\alpha}(em^{-1})^{b+b'\alpha}$. Output m if the condition holds, otherwise output "reject".

Correctness.

Decryption: $eu^{-s} = g^{sr} m g^{-sr} = m$, since $u = g^r$, $e = h^r m$ and $h = g^s$.

Verification: $u^{a+a'\alpha}(em^{-1})^{b+b'\alpha} = (g^r)^{a+a'\alpha}(g^{sr})^{b+b'\alpha} = (g^a h^a (g^{a'} h^{b'})^\alpha)^r$
$= (c(d^\alpha))^r = v$.

Security proof of Short Cramer-Shoup.

Theorem 2. [?]. *The Short Cramer-Shoup (SCS) is IND-PCA under the DDH and the collision-resistance assumptions:*
$Adv_{SCS}^{ind-pca}(t) \leq Adv_{\mathbb{G}}^{ddh}(t) + Succ_{\mathcal{H}}^{coll}(t) + 2(q_p + 1)/p$, *where q_p is the number of queries to the OPCA oracle.*

3 Fast Short Cramer-Shoup

We define three functions: the setup function FSCS.KG(), the encryption function FSCS.Enc() and the decryption function FSCS.Dec().

FSCS.KG(1^λ)**:** Select a cyclic group G of prime order p and a generator g. Pick two random elements $k, q \in \mathbb{Z}_p$ such that the size of q is half of the size of p, i.e., $log_2(q) = \frac{log_2(p)}{2}$. Compute $s', t \in \mathbb{Z}_p$ such that $kp = qs' + t$ and $s \equiv s'(mod\ p)$. Note that the size of t is smaller or equal to the size of q, i.e., $log_2(t) \leq log_2(q)$. Pick four random elements $a, b, a', b' \in \mathbb{Z}_p$. Compute $g_1 = g^s, h = g^t, c = g_1^a h^b, d = g_1^{a'} h^{b'}$. Choose a hash function H that hashes messages to elements of G. Return (pk, sk), where $pk = (g_1, h, c, d, H)$ and $sk = (q, a, b, a', b')$.

FSCS.Enc(pk, m)**:** To encrypt a message m with $pk = (g_1, h, c, d, H)$, choose random element $r \in \mathbb{Z}_p$. Compute $u = g_1^r, e = h^r m, \alpha = H(u, e)$ and $v = (c(d^\alpha))^r$. Output the ciphertext $ct = (u, e, v)$.

FSCS.Dec(sk, ct)**:** To decrypt a ciphertext ct with $sk = (q, a, b, a', b')$. Compute $\alpha = H(u, e)$. Compute $m = eu^q$ and verify if $v = u^{a+a'\alpha}(em^{-1})^{b+b'\alpha}$. Output m if the condition holds, otherwise output "reject".

Correctness.

Decryption: $eu^q = g^{tr} m g^{srq} = mg^{r(sq+t)} = mg^{rkp} = m$, since $u = g^{sr}$, $e = h^r m$ and $h = g^t$.

Verification: $u^{a+a'\alpha}(em^{-1})^{b+b'\alpha} = (g^{sr})^{a+a'\alpha}(h^r)^{b+b'\alpha} = (g_1^a h^b (g_1^{a'} h^{b'})^{\alpha})^r$
$= (c(d^{\alpha}))^r = v$.

IND-PCA Security Proof of Fast Short Cramer-Shoup Scheme. We use the same notions and follows the same proof technique as in [**?**,**?**].

Theorem 3. *The Fast Short Cramer-Shoup (FSCS) is IND-PCA under the DDH and the collision-resistance assumptions:*
$\boldsymbol{Adv}_{FSCS}^{ind-pca}(t) \le \boldsymbol{Adv}_{\mathbb{G}}^{ddh}(t) + \boldsymbol{Succ}_{\mathcal{H}}^{coll}(t) + 2(q_p+1)/p$*, where* q_p *is the number of queries to the OPCA oracle.*

The full proof is given in [**?**] and follows the proof of [**?**].

4 Fast Linear Cramer-Shoup

We define three functions: the setup function, denoted FLCS.KG(), the encryption function, denoted FLCS.Enc() and the decryption function, denoted FLCS.Dec().

FLCS.KG(1^{λ})**:** Choose a random generator $g \xleftarrow{\$} G$ of order p and random elements $k_1, k_2, k_3, q_1, q_2, q_3, x_1, x_2, x_3, y_1, y_2, y_3 \xleftarrow{\$} \mathbb{Z}_p$ such that the size of q_i ($i \in \{1,2,3\}$) is half of the size of p, i.e., $log_2(q_i) = \frac{log_2(p)}{2}$. Compute $s_1, s_2, s_3, t_1, t_2, t_3 \in \mathbb{Z}_p$ such that $k_i p = q_i s_i + t_i$ and $0 < s_i < p$ ($i \in \{1,2,3\}$) and set $b_1 = g^{s_1}$, $b_2 = g^{s_2}$, $b_3 = g^{s_3}$, $c_1 \leftarrow b_1^{x_1} b_3^{x_3}$, $d_1 \leftarrow b_1^{y_1} b_3^{y_3}$, $h_1 \leftarrow g^{t_1+t_3}$, $c_2 \leftarrow b_2^{x_2} b_3^{x_3}$, $d_2 \leftarrow b_2^{y_2} b_3^{y_3}$, $h_2 \leftarrow g^{t_2+t_3}$. The public key is $pk = (b_1, b_2, b_3, c_1, c_2, d_1, d_2, h_1, h_2)$ and the secret key is $sk = (q_1, q_2, q_3, x_1, x_2, x_3, y_1, y_2, y_3)$.

FLCS.Enc(pk, M)**:** To encrypt a message M using pk, choose random exponents $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$, and set $u_1 \leftarrow b_1^{r_1}$, $u_2 \leftarrow b_2^{r_2}$, $u_3 \leftarrow b_3^{r_1+r_2}$ and $e \leftarrow M h_1^{r_1} h_2^{r_2}$. Now compute $\alpha \leftarrow H(u_1, u_2, u_3, e)$ and finally, $v \leftarrow (c_1 d_1^{\alpha})^{r_1} (c_2 d_2^{\alpha})^{r_2}$. The ciphertext is $ct = (u_1, u_2, u_3, e, v) \in G^5$.

FLCS.Dec(pk, sk, ct)**:** Parse the ciphertext ct as $(u_1, u_2, u_3, e, v) \in G^5$. Compute $\alpha \leftarrow H(u_1, u_2, u_3, e)$ and check $u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2} u_3^{x_3+\alpha y_3} \overset{?}{=} v$. If not, output "**reject**". Otherwise, compute and output $M \leftarrow e(u_1^{q_1} u_2^{q_2} u_3^{q_3})$.

Correctness. If the keys and encryption are generated according to the algorithms above, the test in **FLCS.Dec** is satisfied, since we then have

$$\begin{aligned}
v &= (c_1 d_1^{\alpha})^{r_1} (c_2 d_2^{\alpha})^{r_2} \\
&= \left(g^{s_1 x_1 + s_3 x_3} g^{(s_1 y_1 + s_3 y_3)\alpha}\right)^{r_1} \cdot \left(g^{s_2 x_2 + s_3 x_3} g^{(s_2 y_2 + s_3 y_3)\alpha}\right)^{r_2} \\
&= \left(g^{s_1(x_1+\alpha y_1)}\right)^{r_1} \cdot \left(g^{s_2(x_2+\alpha y_2)}\right)^{r_2} \cdot \left(g^{s_3(x_3+\alpha y_3)}\right)^{(r_1+r_2)} \\
&= (b_1^{r_1})^{x_1+\alpha y_1} \cdot (b_2^{r_2})^{x_2+\alpha y_2} \cdot \left(b_3^{(r_1+r_2)}\right)^{(x_3+\alpha y_3)} \\
&= u_1^{x_1+\alpha y_1} \cdot u_2^{x_2+\alpha y_2} \cdot u_3^{x_3+\alpha y_3}
\end{aligned}$$

Next, decryption algorithm recovers the correct M,

$$\begin{aligned} e(u_1^{q_1}u_2^{q_2}u_3^{q_3}) &= M \cdot h_1^{r_1}h_2^{r_2} \cdot b_1^{r_1q_1} \cdot b_2^{r_2q_2} \cdot b_3^{r_3q_3} \\ &= M \cdot g^{(t_1+t_3)r_1}g^{(t_2+t_3)r_2}g^{s_1r_1q_1}g^{s_2r_2q_2}g^{s_3r_3q_3} \\ &= Mg^{r_1(t_1+q_1s_1)} \cdot g^{r_2(t_2+q_2s_2)} \cdot g^{r_3(t_3+q_3s_3)} \\ &= Mg^{r_1k_1p} \cdot g^{r_2k_2p} \cdot g^{r_3k_3p} \\ &= M. \end{aligned}$$

Security proof of Fast Linear Cramer-Shoup (FLCS). We now show that the the FLCS scheme is CCA secure.

Theorem 4. *The FLCS scheme is secure in the CCA sense if $\mathcal{HF}$ a secure* ***UOWHF*** *family and the Linear assumption hold in G.*

The full proof is given in [**?**] and follows the proof of [**?**].

5 Performances Evaluation

We compare efficiency between our proposed schemes and existing ones. We first study the complexity and the performance of the short Cramer-Shoup variant, namely *Fast Short Cramer-Shoup* encryption scheme (Section 3) and *Short Cramer-Shoup* encryption scheme (Section 2.3). Next, we study the complexity and the performance of the linear construction, namely *Fast Linear Cramer-Shoup* encryption scheme (Section 4) with *Linear Cramer-Shoup* encryption scheme (Section 2.2).

In both cases (short and linear variants), we chose to compare them algorithm by algorithm. Hence, we study key generation, encryption and decryption algorithms apart. Note that the decryption algorithm is composed of two steps: a verification and the actual decryption (for retrieving the initial message). Thus, the full decryption algorithm is divided in two, each part corresponding to those specific phases (verification and actual decryption).

For all algorithms, we split the study in two approaches to conduct such comparison. The first one is relative to the theoretical complexity; we look the number of operations needed for each algorithm. The second one is an experimental study. For this, we have implemented the schemes using the C-library GMP [**?**] for computing the average execution time of algorithms. In all schemes, there are 1000 execution trials where new security parameters and messages are randomly generated for each execution. For a complete comparison though, the security parameters (such as prime number) and messages are the same for the schemes. The curves shown are the average execution time for a given size of security parameter (from $2^9 = 512$ to $2^{12} = 4096$ bits). Our proposed schemes are always represented by (black) circle points whereas standard schemes (Linear CS and Short CS) are represented by (blue) square points.

	Public Key Parameters	
Number of	Short CS	Fast Short CS
Elements	4	4
	Secret Key Parameters	
Elements	5	5

Table 1. Comparison of Short and Fast Short Cramer-Shoup for key parameters.

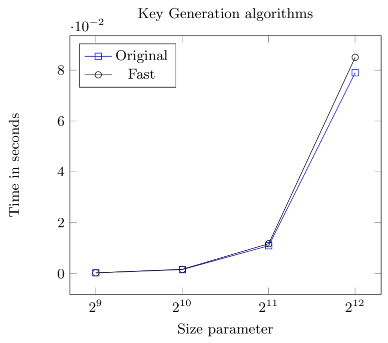

Fig. 1. Key Generation comparison of Short and Fast Cramer-Shoup.

5.1 Short and Fast Short Cramer-Shoup

Key Generation Algorithms. We look for the differences between the key generation algorithm of *Fast Short CS* (Section 3) and *Short CS* protocol (Section 2.3). Table 1 shows that our scheme has the same number of parameters in the public and secret keys. Table 2 gives the number of parameters needed in this phase. The most noticeable difference lies in the number of modular exponentiations. Indeed, the short version uses only 5 of them while our uses 6. The additional exponent comes from the term $g_1 = g^s$; our construction implies to use this element instead of a simple generator (as in the standard version). This computation's difference can be observed in Fig. 1, as expected. We conclude that key generation is slightly slower for our proposed scheme. However,

	Key Generation	
Number of	Short CS	Fast Short CS
Generator	1	1
Random	**5**	7
Multiplication	2	2
Exponentiation	**5**	6

Table 2. Comparison of Short and Fast Short Cramer-Shoup. We emphasize the minimum for each row with bold.

	Encryption	
Number of	Short CS	Fast Short CS
Random	1	1
Multiplication	2	2
Exponentiation	4	4

Table 3. Comparison of Short and Fast Short Cramer-Shoup for encryption.

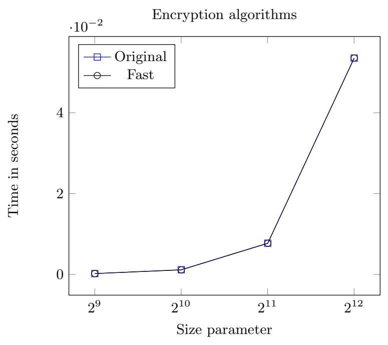

Fig. 2. Encryption comparison of Short and Fast Cramer-Shoup.

this inconvenient will be greatly rewarded during the decryption algorithm. Note that the key generation algorithm is ran only once per party thus the balance is in favour of the *Fast Short Cramer-Shoup* if several messages are sent/received with the same pair of key (i.e., the practical case).

Encryption Algorithms. We now study the encryption algorithm. Since both schemes use the same encryption algorithm, we have the same number of operations, as it is shown in Table 3. This matches with the average execution time given in Fig. 2.

Decryption Algorithms. Our contribution lies on a faster decryption algorithm. The average execution time is given in Fig. 3.

The decryption algorithms are composed of two distinct phases: a verification to check integrity of the message sent, and the actual decryption where the message is decrypted. Note that the full decryption algorithm from the short

	Decryption	
Number of	Short CS	Fast Short CS
Inverse	1	**0**
Multiplication	1	1
Exponentiation	1	1

Table 4. Comparison of Short and Fast Short Cramer-Shoup for decryption.

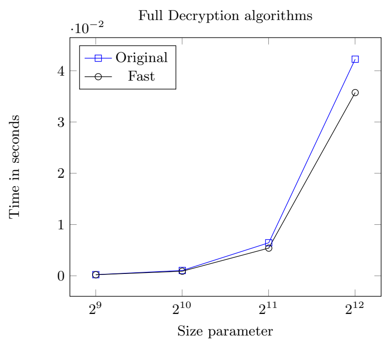

Fig. 3. Full decryption comparison of Short and Fast Cramer-Shoup.

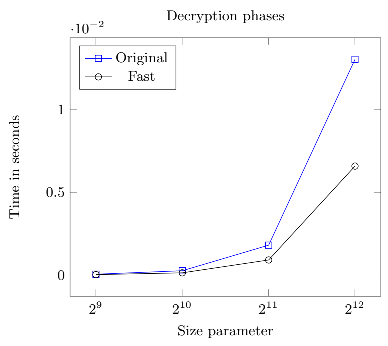

Fig. 4. Comparison of Short and Fast Cramer-Shoup for the actual decryption.

variant of Cramer-Shoup is reversed toward the standard Cramer-Shoup. The actual decryption is done first then the verification is computed from the message previously retrieved.

Actual Decryption. Our construction is dedicated to improve the actual decryption. There are two explanations for understanding the improvement of the average execution time (Fig. 4) during this phase. Firstly, The number of multiplication and modular exponentiation are the same, but the number of operations is reduced for the *Fast Short CS*. As depicted in Table 4, there is no inverse computation while the *Short CS* needs one. The second explanation lies on the modular exponentiation itself (from a purely computational point of view).

Indeed, despite the fact that both algorithms have the same computation there is a major difference, namely the size of the exponent. In the *Fast Short CS*, the exponent q has its size half of the security parameter leading to a faster modular exponentiation.

Verification phase. Both schemes have the same verification computations thus we have the same average execution time as shown in Fig. 5.

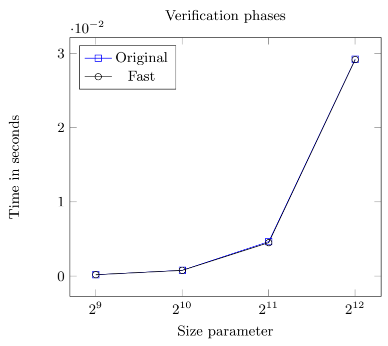

Fig. 5. Comparison of Short and Fast Cramer-Shoup for verification.

Verification		
Number of	Short CS	Fast Short CS
Inverse	1	1
Multiplication	2	2
Exponentiation	2	2

Table 5. Comparison of Short and Fast Short Cramer-Shoup for verification.

5.2 Linear and Fast Linear Cramer-Shoup

We study the complexity and average execution time of the algorithms of *Linear CS* and *Fast Linear CS*. We compare the key generation algorithms of *Linear CS* and *Fast Linear CS*. From Table 6, we can see that there is one less modular exponentiation in the standard scheme. However, the fast version has two exponentiations: $h_1 = g^{t_1+t_3}$ and $h_1 = g^{t_2+t_3}$, where elements t_i are computed as the rest of the euclidean division (recall that the equations are : $k_i p = q_i s_i + t_i$ for $i = 1, 2, 3$). We have $t_i \leq q_i$ where the size of q_i is the half of the size of p. Thus elements t_i have in average a size half of the size of q_i leading to smaller exponentiation of h_1 and h_2 in the fast version. In addition, the fast variant has two less multiplications than the standard scheme. The results of our experiences,

Key Generation		
Number of	Linear CS	Fast Linear CS
Generator	3	**1**
Random	9	9
Multiplication	6	**4**
Exponentiation	**12**	13

Table 6. Key Generation comparison of Linear and Fast Linear Cramer-Shoup.

Public Key Parameters		
Number of	Linear CS	Fast Linear Fast CS
Elements	9	9
Secret Key Parameters		
Elements	9	9

Table 7. Key Parameters comparison of Linear and Fast Linear Cramer-Shoup.

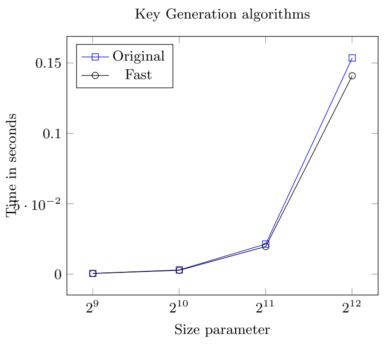

Fig. 6. Key Generation comparison of Linear and Fast Linear Cramer-Shoup.

presented in Fig. 6, confirm this slight improvement. As shown in Table 7 both schemes have the same number of key parameters.

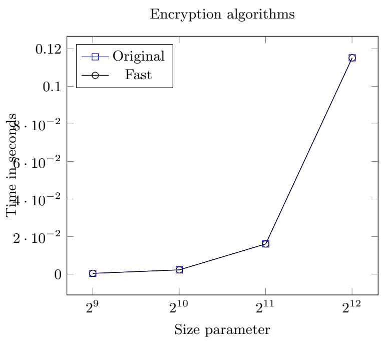

Fig. 7. Encryption comparison of Linear and Fast Linear Cramer-Shoup.

Encryption		
Number of	Linear CS	Fast Linear CS
Random	2	2
Multiplication	5	5
Exponentiation	9	9

Table 8. Comparison of Linear and Fast Linear Cramer-Shoup for encryption.

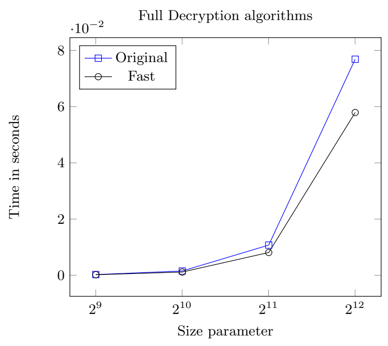

Fig. 8. Full Decryption comparison of Linear and Fast Linear Cramer-Shoup.

Verification

Number of	Linear CS	Fast Linear CS
Multiplication	2	2
Exponentiation	3	3

Table 9. Verification comparison of Linear and Fast Linear Cramer-Shoup.

Encryption Algorithms. Both schemes use the same encryption algorithm thus the number of operations (Table 8) is the same so as the average execution time (Fig. 7).

Decryption Algorithms. We observe in Fig. 8 that our proposed scheme has a faster decryption algorithm.

Verification phase. The verification is identical in both schemes. Hence they have same execution time. The results given in Table 9 and Fig. 9 corroborate it.

Actual Decryption. The construction of the *Fast Linear CS* aims at reducing the execution time of this phase. In Table 10, we observe that the number of multiplication and modular exponentiation are the same. However, there is no inverse computation in the fast version unlike the standard scheme. This is the first ex-

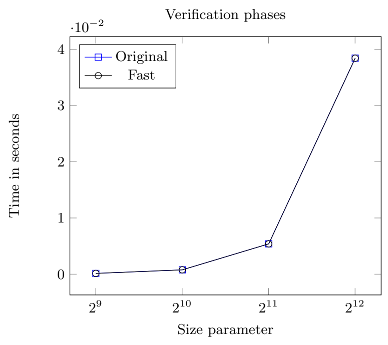

Fig. 9. Verification comparison of Linear and Fast Linear Cramer-Shoup.

	Decryption	
Number of	Linear CS	Fast Linear CS
Inverse	3	**0**
Multiplication	3	3
Exponentiation	3	3

Table 10. Comparison of Linear and Fast Linear Cramer-Shoup for decryption.

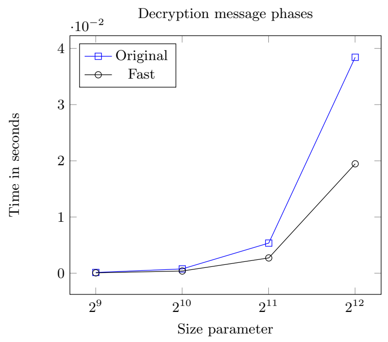

Fig. 10. Actual decryption Comparison of Linear and Fast Linear Cramer-Shoup.

planation for the result given in Fig. 10. This cannot be the only reason yet since the *Fast Linear CS* decryption is about twice as fast as the *Linear CS*. Indeed, the second explanation for such result concerns the modular explanation itself. Recall the decryption computations of the schemes: LCS: $M = e/(u_1^{z_1} u_2^{z_2} u_3^{z_3})$ and Fast LCS: $M = e(u_1^{q_1} u_2^{q_2} u_3^{q_3})$. The exponents z_1, z_2, z_3 of standard scheme are drawn from $\mathbb{Z}_p$ while the exponents q_1, q_2, q_3 of fast version have their size equal to the half of the security parameter. Hence, in average, the modular exponentiation costs less from the latter elements. This conclude the study of the actual decryption where our proposed scheme needs only half of the execution time of the standard scheme. Yet, this important gain is relative to the full decryption algorithm where verification phase constitutes the majority of the execution time.

6 Conclusion

We designed two schemes to improve Short and Linear Cramer-Shoup schemes. We prove the same security as the original schemes for our faster schemes and under the same hypothesis. We also confirm experimentally the significant gain in our decryption algorithms.

References

1. M. Abdalla, F. Benhamouda, and D. Pointcheval. Public-key encryption indistinguishable under plaintext-checkable attacks. Cryptology ePrint Archive, Report 2014/609, 2014.
2. M. Abdalla, F. Benhamouda, and D. Pointcheval. Public-key encryption indistinguishable under plaintext-checkable attacks. In J. Katz, editor, *Public-Key Cryptography – PKC 2015*, pages 332–352, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.
3. anonymous. Full version our paper, 2020. `https://drive.google.com/file/d/1V-RwpHLK-sFCCOOU-QNAgwFD-sjbLVGV/view?usp=sharing`.
4. D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Computing*, 32(3):586–615, 2003.
5. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Proc. of the 18th Annual International Cryptology Conference on Advances in Cryptology, crypto'98*, pages 13–25, 1998.
6. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. Cryptology ePrint Archive: Report 1998/006, march 1998.
7. B. L. D. Boneh and H. Shacham. Short signatures from the weil pairing. *J. Cryptology*, 17(4):297–319, Sept. 2004.
8. X. B. D. Boneh and H. Shacham. Short group signatures. In *In M. Franklin, editor,Proceedings of Crypto 2004*, volume 3152, pages 41–55, Aug. 2004.
9. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO, IT-31(4)*, volume 4, pages 469–472, 1985.
10. T. Granlund and the GMP development team. *GNU MP: The GNU Multiple Precision Arithmetic Library*, 6.2.0 edition, 2020.
11. A. Joux and K. Nguyen. Separating decision diffie-hellman from computational diffie-hellman in cryptographic groups. *J. Cryptology*, 16(4):239–47, Sept. 2003.
12. M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. ACM Press, May 1989.
13. K. Paterson. Cryptography from pairings. *Cambridge University Press*, 317 of London Mathematical Society Lecture Notes:215–51, 2005.
14. H. Shacham. A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074, 2007.
15. V. Shoup. Lower bounds for discrete logarithms and related problems. In *In W. Fumy, editor, Proceedings of Eurocrypt 1997*, volume 1233 of LNCS, pages 256–66, May 1997.
16. D. Sow and D. Sow. A new variant of el gamal's encryption and signatures schemes. *JP Journal of Algebra, Number Theory and Applications*, 20(1):21–39, 2011.