Non-committing encryption from Φ-hiding

Brett Hemenway Rafail Ostrovsky* Alon Rosen†

January 22, 2015

Abstract

A multiparty computation protocol is said to be adaptively secure if it retains its security even in the presence of an adversary who can corrupt participants as the protocol proceeds. This is in contrast to the static corruption model where the adversary is forced to choose which participants to corrupt before the protocol begins.

A central tool for constructing adaptively secure protocols is non-committing encryption (Canetti, Feige, Goldreich and Naor, STOC '96). The original protocol of Canetti et al. had ciphertext expansion that was quadratic in the security parameter, and prior to this work, the best known constructions had ciphertext expansion that was linear in the security parameter.

In this work, we present the first non-committing encryption scheme that achieves ciphertext expansion that is logarithmic in the message length. Our construction has optimal round complexity (2-rounds), where (just as in all previous constructions) the first message consists of a public-key of size $\tilde{\mathcal{O}}(n\lambda)$ where n is the message length and λ is the security parameter. The second message consists of a ciphertext of size $\mathcal{O}(n \log n + \lambda)$. The security of our scheme is proved based on the Φ-hiding problem.

*The work of R. Ostrovsky was supported in part by NSF grants CCF-0916574, IIS-1065276, CCF-1016540, CNS-1118126, CNS-1136174; US-Israel BSF grant 2008411; OKAWA Foundation Research Award; IBM Faculty Research Award; Xerox Faculty Research Award; B. John Garrick Foundation Award; Teradata Re- search Award; and Lockheed-Martin Corporation Research Award. This material is also based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0392.

†Work supported by ISF grant no. 1255/12 and by the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 307952. Part of this work done while visiting UCLA.

1 Introduction

Secure multiparty computation (MPC) allows a group of players to compute any joint function of their inputs while maintaining the privacy of each individual player's input. When defining security of an MPC protocol, it is important to consider how and when players are corrupted. Traditionally security has been considered in two adversarial models, *static* and *adaptive*. In the static model, the corrupted players are fixed before the protocol begins, while in the adaptive model players may become corrupted at any point in the protocol. In particular, the adversary may choose to corrupt players adaptively (up to a certain threshold) based on messages sent in earlier rounds of the protocol. The adaptive security model, which more accurately reflects conditions in the real-world, is widely recognized to be "the right" model for security in the MPC setting. Unfortunately, analysis of protocols in the adaptive model can be difficult, and many protocols are only proven secure in the weaker static corruption model.

The Goldreich, Micali, Wigderson protocol [14] provides computational security in the static model, and it was observed in [11] that the original GMW proof of security does not hold in the adaptive corruption model. The works of Ben-Or, Goldwasser and Wigderson [3] and Chaum, Crépeau and Damgård [7] take an alternate approach to MPC and create protocols with information theoretic security. These protocols rely on a majority on private channels between the players, and require a majority of players to be honest. Like the GMW protocol, the BGW and CCD protocols were only proven secure in the static corruption model.

1.1 Non-Committing Encryption

In [5], Canetti, Feige, Goldreich and Naor introduced the notion of *non-committing encryption*, and showed that if the private channels in the BGW and CCD protocols are replaced by non-committing encryption, then the MPC protocols can achieve adaptive security. Subsequent work has made similar use of non-committing encryption in order to construct cryptographic protocols achieving security against adaptive adversaries [1, 6].

Despite its utility, previous constructions of non-committing encryption have been much less efficient than standard (IND-CPA secure) cryptosystems. The original construction of Canetti et. al. created non-committing encryption from *common-domain* trapdoor permutations, which in turn could be realized from either the CDH or RSA assumption. The resulting scheme requires $\mathcal{O}(\lambda)$ ciphertexts (each of bit length $\mathcal{O}(\lambda)$) to encrypt a single bit plaintext with security parameter λ.

Beaver [1] constructed a non-committing key-exchange protocol based on the DDH assumption, but his protocol requires 3 rounds. Beaver's construction provided an *interactive* encryption protocol with ciphertext expansion $\mathcal{O}(\lambda)$. Damgård and Nielsen [11] demonstrated a generalization of Beaver's scheme that could be implemented under general assumptions. The protocol of Damgård and Nielsen, like that of Beaver, requires three rounds and has ciphertext expansion $\mathcal{O}(\lambda)$. The construction of Damgård and Nielsen was later improved, reducing the round complexity to two rounds, by Choi, Dachman-Soled, Malkin and Wee [8]. The construction of [8] gives a two-round protocol with ciphertext expansion $\mathcal{O}(\lambda)$ and public keys of size $\mathcal{O}(n\lambda)$ for n-bit messages. Achieving non-interactive (2-round) non-committing encryption with optimal rate (i.e., $\mathcal{O}(1)$ ciphertext expansion) remained an open question.

1.2 Our Contributions

We construct the first non-committing encryption scheme with logarithmic ciphertext expansion. Our scheme is 2-round, which is optimal in terms of round complexity. The security of our construction rests on the Φ-hiding assumption [4]. Our construction improves on the ciphertext size of previous constructions, with only a modest cost in key size (our public key sizes are $\tilde{\mathcal{O}}(n\lambda)$, compared to $\mathcal{O}(n\lambda)$ in [8]).

All prior constructions of non-committing encryption are loosely based on the following paradigm. The receiver will generate a set of n public-keys for which he only knows the decryption keys corresponding to a small subset, I_R. The sender will first encode his message using an error-correcting code of block-length n, then the sender will choose a small subset of keys, I_S, and encrypt one symbol of the codeword under keys in I_S and sample ciphertexts obliviously for the remaining keys. The parameters then need to be tuned so that if the sender's and receiver's sets are chosen independently, the intersection $I_R \cap I_S$ will be large enough to recover the message with high probability. On the other hand, the simulator (who knows decryption keys

Reference	Rounds	Ciphertext Expansion	Assumption
[5]	2	$\mathcal{O}(\lambda^2)$	Common-Domain TDPs
[1]	3	$\mathcal{O}(\lambda)$	DDH
[11]	3	$\mathcal{O}(\lambda)$	Simulatable PKE
[8]	2	$\mathcal{O}(\lambda)$	Simulatable PKE
This work	2	$\mathcal{O}(\log \lvert n \rvert)$	Φ-hiding

Figure 1: Comparison to prior work. The parameter λ denotes the security parameter, and $|n|$ denotes the message length. Notice that $\lambda \gg \log |n|$ since by assumption super-polynomial time calculations in λ are infeasible, while exponential-time calculation in $\log |n|$ are feasible.

for all public-keys, and encryption randomness for all ciphertexts) should be able to find a subset of secret keys, and a subset of encryption randomness, such that the ciphertexts in their intersection decode to any target message.

In these schemes, each of the underlying encryptions encrypts one symbol from an error-correcting encoding of the sender's message. Let Σ denote the alphabet of the error-correcting code. Then, even if the simulator knows decryptions for every one of the n ciphertexts, decryptions will only agree with a random vector in Σ^n in a $\frac{1}{|\Sigma|}$ fraction of coordinates. Thus if Σ is large, the simulator's job of finding an intersection $I_R \cap I_S$ that matches the target message will be impossible. On the other hand, if Σ is small then the scheme suffers from large ciphertext expansion because each codeword symbol is encrypted separately, and the semantic security of the underlying cryptosystem puts a lower bound on the size of each ciphertext. This has been a fundamental barrier to achieving non-committing encryption with good rate.

In order to overcome this barrier, we design a cryptosystem where randomness can be re-used across keys. Cryptosystems with this property are easy to construct, and even El-Gamal has this property. In fact, cryptosystems coming from smooth hash proof systems [10] have this property, as do cryptosystems based on the extended decisional diffie-hellman assumption [16]. This notion of randomness re-use across keys was a necessary ingredient in constructing lossy trapdoor functions [20] and multi-recipient cryptosystems [2].

Now, if we have n ciphertexts, but each uses the same randomness, the total length could be as small as $\mathcal{O}(n + \lambda)$ instead of $\mathcal{O}(n\lambda)$. Unfortunately, reusing randomness across ciphertexts makes the simulator's job much harder. Now, when the simulator generates encryption randomness, it must efficiently generate a *single* random element that is simultaneously valid randomness for all n ciphertexts. For a given cryptosystem, it can be difficult to determine whether such randomness exists, and more difficult still to devise an *efficient* scheme for finding it.

Although we can build randomness-reusable cryptosystems from almost all standard hardness assumptions, (e.g. DDH, DCR, QR) in order to achieve the efficient randomness sampling required by the simulator, we rely on the Φ-hiding assumption [4].

At a high level, our non-committing encryption behaves as follows. The receiver generates n public-keys for a randomness-reusable cryptosystem based on the Φ-hiding assumption, in such a way that he only knows the decryption keys for a small subset I_R. The sender encodes his message using a binary error-correcting code of block-length n, and chooses a subset of keys $I_S \subset [n]$. The sender generates a single piece of randomness and encodes the symbols of his codeword using this single random value for $i \in I_S$, and the sender samples ciphertexts obliviously otherwise.

Encryption under our scheme is conceptually similar to all previous constructions of non-committing encryption, but re-using randomness makes creating an efficient simulator significantly more difficult. On the positive side re-using randomness allows us to decrease the ciphertext size beyond the bounds achieved compared to all previously known constructions.

2 Preliminaries

2.1 Notation

If A is a Probabilistic Polynomial Time (PPT) machine, then we use $a \xleftarrow{\$} A$ to denote running the machine A and obtaining an output, where a is distributed according to the internal randomness of A. If R is a set,

we use $r \xleftarrow{\$} R$ to denote sampling uniformly from R.

We use the notation

$$\Pr[A(x,r) = c : r \xleftarrow{\$} R, x \xleftarrow{\$} X],$$

to denote the probability that A outputs c when x is sampled uniformly from X and r is sampled uniformly from R. We define the statistical distance between two distributions X, Y to be

$$\Delta(X,Y) = \frac{1}{2}\sum_x |\Pr[X = x] - \Pr[Y = x]|$$

If X and Y are families of distributions indexed by a security parameter λ, we use $X \approx_s Y$ to mean the distributions X and Y are statistically close, *i.e.*, for all polynomials p and sufficiently large λ, we have $\Delta(X,Y) < \frac{1}{p(\lambda)}$. We use $X \approx_c Y$ to mean X and Y are computationally close, *i.e.*, for all PPT adversaries A, for all polynomials p, then for all sufficiently large λ, we have $|\Pr[A^X = 1] - \Pr[A^Y = 1]| < 1/p(\lambda)$.

2.2 Non-Committing Encryption

In this section, we review the notion of non-committing encryption [5]. Essentially, a non-committing encryption scheme is one where there is an efficient simulator, that can generate public keys and ciphertexts, such that, at a later point, a corresponding secret key and encryption randomness can be generated which "opens" the ciphertext to any given message. For security, we require that the distribution on simulated keys and ciphertexts is computationally indistinguishable from the distribution on keys and ciphertexts generated by the real encryption protocol. The formal definition is given in Definition 1.

Definition 1 (Non-Committing Encryption)**.**
A cryptosystem $\mathcal{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is called non-committing, if there exists a simulator Sim such that the following properties hold:

1. **Efficiency:** The algorithms $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ and Sim are all probabilistic polynomial time.
2. **Correctness:** For any message $m \in \mathcal{M}$

$$\Pr[\mathsf{Dec}(sk,c) = m : (pk,sk) \xleftarrow{\$} \mathsf{Gen}(1^\lambda), c \xleftarrow{\$} \mathsf{Enc}(pk,m)] = 1$$

3. **Simulatability:** The simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ For any message $m \in \mathcal{M}$, define the distributions:

$$\Lambda_m^{\mathrm{Sim}} = \{(m, pk, sk, r_1, r_2) : (pk, c, t) \xleftarrow{\$} \mathsf{Sim}_1(1^\lambda), (sk, r_1, r_2) \xleftarrow{\$} \mathsf{Sim}_2(m, t)\}$$

and

$$\Lambda_m^{\mathrm{Real}} = \{(m, pk, sk, r_1, r_2) : (pk, sk) \xleftarrow{\$} \mathsf{Gen}(1^\lambda; r_1), c \xleftarrow{\$} \mathsf{Enc}(pk, m; r_2)\}$$

We require that $\Lambda_m^{\mathrm{Sim}} \approx_c \Lambda_m^{\mathrm{Real}}$.

We do not explicitly require that the system be semantically secure, this is implied by the simulatability of the system.

2.3 The Φ-hiding Assumption

The security of our construction rests on the Φ-hiding assumption [4]. Informally, the Φ-hiding assumption asserts that it is hard to determine whether a given integer, e, divides the size of the group $\mathbb{Z}_N^*$ when $N = pq$ is an RSA modulus.

Let $\mathsf{PRIMES}(\lambda)$ denote the set of primes of bit-length λ, and let $\mathsf{RSA}(\lambda)$ denote the set of RSA moduli of bit-length λ, i.e.,

$$\mathsf{RSA}(\lambda) = \{N : N = pq, p, q \in \mathsf{PRIMES}(\lambda/2), \gcd(p-1, q-1) = 2\}$$

Let $\mathsf{RSA}_e(\lambda)$ denote the set of RSA moduli of length λ that Φ-*hide* e.

$$\mathsf{RSA}_e(\lambda) = \{N \in \mathsf{RSA}(\lambda) : e \text{ divides } p-1\}$$

The Φ-hiding assumption states that a random sample from $\mathsf{RSA}_e(\lambda)$ is computationally indistinguishable from a random sample from $\mathsf{RSA}(\lambda)$.

Definition 2 (The Φ-hiding assumption)**.**
For all $\epsilon > 0$, and for all $e \in \mathbb{Z}$ such that $3 < e < 2^{\lfloor \log(\lambda/4) \rfloor - \epsilon}$ and for all PPT distinguishers, A

$$\left| \Pr[A(N, e) : N \xleftarrow{\$} \mathsf{RSA}(\lambda)] - \Pr[A(N, e) : N \xleftarrow{\$} \mathsf{RSA}_e(\lambda)] \right| < \nu(\lambda)$$

For some negligble function $\nu(\cdot)$.

The original Φ-hiding assumption stated that $\mathsf{RSA}_e(\lambda)$ and $\mathsf{RSA}_{e'}(\lambda)$ are indistinguishable for all $3 < e \leq e' < 2^{\lambda/5}$. For simplicity, we use a slightly modified assumption (i.e., that $\mathsf{RSA}_e(\lambda) \approx_c \mathsf{RSA}(\lambda)$). All of our constructions go through essentially unchanged under the original Φ-hiding assumption, except that the key generator would have to choose an e to sample from $\mathsf{RSA}_e(\lambda)$ and then ignore e throughout the protocol.

The bound $e < 2^{\lfloor \log(\lambda/4) \rfloor - \epsilon}$ is necessary to avoid Coppersmith's attack [9]. The Φ-hiding assumption was originally used to build efficient Private Information Retrieval (PIR) protocols [4, 13]. In [17] it was observed that the Φ-hiding assumption turns the RSA map $x \mapsto x^e \mod N$ into a lossy trapdoor function (LTF) [20]. Kiltz, O'Neill and Smith leveraged the lossiness to remove the random oracle from the proof of security of RSA-OAEP. More recently, the regularity of the lossy RSA map has been explored [18].

3 Smooth Hash Proofs from Φ-hiding

Our work builds on the notion of smooth hash proof systems as defined by Cramer and Shoup [10]. A hash proof system for a language, L, is a set of keyed hash functions H_k, along with "public" and "private" evaluation algorithms for H_k. A hash function H_k (for $k \in K$) takes $X \to \Pi$.

The public algorithm takes $x \in L$, along with a witness w, and the "projected key" $\alpha(k)$, and outputs $H_k(x)$. The private evaluation algorithm simply takes x, k and outputs $H_k(x)$. While the public and private evaluations must match on $x \in L$, if $x \notin L$, the smoothness property says that $H_k(x)$ should be almost uniformly distributed (even conditioned on $\alpha(k)$). Formally, we have

Definition 3 (Hash Proof Systems [10])**.** The set $(H, K, X, L, \Pi, S, \alpha)$ is a *hash proof system* if, for all $k \in K$, the action of H_k on the subset L is completely determined by $\alpha(k)$.

Thus for a hash proof system the projection key $\alpha(k)$ determines the action of H_k on L. We also require that given $x \in L$, and a witness w, along with the projection key $\alpha(k)$ then the hash value $H_k(x)$ can be *efficiently* computed. This is called the *public evaluation algorithm.*

A hash proof system is *smooth* if the projected key $\alpha(k)$ encodes almost no information about the action of $H_k(\cdot)$ outside L.

Definition 4 (Smooth Hash Proof Systems [10])**.** Let $(H, K, X, L, \Pi, S, \alpha)$ be a hash proof system, and define two distributions Z_1, Z_2 taking values on the set $X \setminus L \times S \times \Pi$. For Z_1, we sample $k \leftarrow K$, $x \leftarrow X \setminus L$, and set $s = \alpha(k)$, $\pi = H_k(x)$, for Z_2 we sample $k \leftarrow K$, $x \leftarrow X \setminus L$, and $\pi \leftarrow \Pi$, and set $s = \alpha(k)$. The hash proof system is called ν-*smooth* if $\Delta(Z_1, Z_2) < \nu$.

We will call an infinite family of hash proof systems smooth if the family is ν-smooth for some negligible function ν. For a language, L with no polynomial-time distinguisher, smooth hash proof systems immediately imply IND-CPA secure encryption by letting k be the secret key, $\alpha(k)$ be the public key, and to encrypt a message $m \in \Pi$ you sample $x \in L$ along with a witness and use the public evaluation algorithm to output $E(m) = (x, H_k(x) + m)$.

We begin by describing a simple smooth hash proof system from the Φ-hiding assumption. Building a hash proof system based on Φ-hiding is straightforward, because Φ-hiding essentially implies that the discrete-log problem is hard[1], and then we can used the framework for the original discrete-log based hash proof systems from [10].

[1]Suppose g is a generator for the largest cyclic component of $\mathbb{Z}_N^*$ and let $h = g^p \mod N$. If N Φ-hides p, then h most elements of $\mathbb{Z}_N^*$ will *not* be powers of h. On the other hand, if $\gcd(p, \varphi(N)) = 1$, then h will generate the largest cyclic component of $\mathbb{Z}_N^*$. If we had an oracle that could find the discrete-logarithm (with base h) in $\mathbb{Z}_N^*$, then we could distinguish these two cases. For the same reason, the Φ-hiding assumption also implies that the determining membership in the subgroup of pth powers is hard.

- **Key Generation:** Choose a strong $(\tau, 2^{-\lambda})$-extractor[2] $\mathrm{Ext} : \{0,1\}^d \times \mathbb{Z}_N \to \{0,1\}$ Fix a prime p with $2^\tau < p < N^{\frac{1}{4}}$. Sample $N \xleftarrow{\$} \mathsf{RSA}_p(\lambda)$ Sample $r \xleftarrow{\$} \mathbb{Z}_N$. Set $g = r^p \in \mathbb{Z}_N$.

  Sample an extractor seed, $\mathfrak{s} \xleftarrow{\$} \{0,1\}^d$.

  The language will be the set of pth powers in $\mathbb{Z}_N$.[3]

  Sample $k \xleftarrow{\$} \mathbb{Z}_N$ Set the projection key, $\alpha(k) = g^k \mod N$.

- **Public Evaluation:** Given $x = g^w \mod N$, and a witness, w,

$$H_k(x) = \mathrm{Ext}_{\mathfrak{s}}\left((g^k)^w \mod N\right)$$

- **Private Evaluation:** Given the secret key, k, and a point x,

$$H_k(x) = \mathrm{Ext}_{\mathfrak{s}}\left(x^k \mod N\right)$$

Correctness is easy to verify. To see smoothness, suppose N Φ-hides p. Then, $\mathbb{Z}_N^*$ has a subgroup of order p, but $g = r^p \mod N$ has no component in that subgroup. In particular, $h = g^k \mod N$, only depends on $k \mod \varphi(N)/p$.

Now, if x has full order in the largest cyclic subgroup[4] of $\mathbb{Z}_N^*$ then, assuming $\gcd(p, w) = 1$, the value $(x^w)^k \bmod N$ depends on all of k, and in particular has $\log p$ bits of entropy even conditioned on $k \mod \varphi(N)/p$ (and hence the same holds conditioned on g^x). Thus the extractor produces an almost uniform value, even conditioned on the projected key $g^k \mod N$.

3.1 A compact hash proof system from the Φ-hiding assumption

In this section, we build a simple IND-CPA secure cryptosystem that will be the foundation of our full non-committing construction.

- **Key Generation:** Let ℓ be the (bit) length of messages that will be encrypted. Choose a strong $(\tau, 2^{-\lambda})$-extractor[5] $\mathrm{Ext} : \{0,1\}^d \times \mathbb{Z}_N \to \{0,1\}$

  Let $p_i =$ $(i+2)$nd prime,

$$(p_1, p_2, p_3, \ldots) = (5, 7, 11, \ldots)$$

  Let $e_i = \lceil \tau \log_{p_i} 2 \rceil$ for $i = 1, \ldots, \ell$. Let $q_i = p_i^{e_i}$, thus $q_i > 2^\tau$.

  Sample $N \xleftarrow{\$} \mathsf{RSA}(\lambda)$.

  Sample an extractor seed, $\mathfrak{s} \xleftarrow{\$} \{0,1\}^d$. Sample $a_1, \ldots, a_\ell \xleftarrow{\$} \mathbb{Z}_N$. Set

$$h_i = \left(g^{\prod_{j \neq i} q_i}\right)^{a_i} \mod N$$

  for $i = 1, \ldots, \ell$.

- **Encryption:** Given a message $\vec{m} = (m_1, \ldots, m_\ell) \in \{0,1\}^\ell$

  Pick $w \xleftarrow{\$} \mathbb{Z}_N$

$$\vec{c} = (g^w \mod N, \mathfrak{s}, c_1, \ldots, c_\ell)$$

  where

$$c_i = \mathrm{Ext}_{\mathfrak{s}}\left((h_i)^w\right) \oplus m_i$$

[2] In other words, Ext, if X is a random variable supported on $\mathbb{Z}_N$ with min-entropy at least τ, then if $\mathfrak{s} \xleftarrow{\$} \{0,1\}^d$ the statistical distance between $(\mathfrak{s}, \mathrm{Ext}_{\mathfrak{s}}(X))$ and $(\mathfrak{s}, U_{\{0,1\}})$ is less than $2^{-\lambda}$.

[3] Note that if N Φ-hides p then $\mathbb{Z}_N^*$ has a cyclic subgroup of order p, and so at most $\varphi(N)/p$ residues in $\mathbb{Z}_N^*$ that are pth powers, otherwise all elements in $\mathbb{Z}_N^*$ will be pth powers.

[4] If we choose N to be the product of two primes, N_1, N_2 and $\gcd(N_1 - 1, N_2 - 1) = 2$, then we have that $\mathbb{Z}_N^*$ has a cyclic subgroup of order $\varphi(N)/2 = (N_1 - 1)(N_2 - 1)/2$, and we can work in this group.

[5] The explicit extractors of [15] give seed length $d = \mathcal{O}(\log\log N + \log 1/\epsilon)$ and can extract $\omega(k)$ bits where k is the min-entropy of the source. In our construction, we will only need to extract one bit.

- **Decryption:** Notice that

$$(h_i)^w = \left((g^w)^{\Pi_{j\neq i} q_j}\right)^{a_i}$$

Thus the receiver, who has a_i, can recover m_i by setting

$$m_i = \mathrm{Ext}_{\mathfrak{s}}\left(\left((g^w)^{\Pi_{j\neq i} q_j}\right)^{a_i}\right) \oplus c_i$$

The IND-CPA security of this scheme follows immediately from the security of the underlying hash proof system.

Remark 1: Since messages must be polynomial in the security parameter, λ, we have that $\ell \log \ell < \sqrt[5]{N} \approx 2^{\lambda/5}$, so the conditions of the Φ-hiding assumption are satisfied.

Remark 2: Although the security follows from the Φ-hiding assumption, the honest receiver does not need to choose a modulus that Φ-hides anything.

Remark 3: Notice that the factorization of N is never used, and need not be known for decryption.

4 Non-committing encryption from Φ-hiding

The high level idea is as follows: to encrypt a message m the sender encodes the message m using a standard binary error correcting code. The sender will then flip a small number of bits of the codeword. The sender then encrypts each bit of this corrupted codeword separately (using different public-keys, but the same encryption randomness).

The receiver will know only a subset of the total set of possible secret keys. The receiver will decode only the bits corresponding to the keys he knows. With high probability, the overlap between the receiver's decryption keys and the bits of the codeword flipped by the sender will be small enough to be handled by the error correcting code. The formal scheme is given in Figure 2.

4.1 Probability of Decryption Error

First, we examine the probability of decryption error. It is a simple calculation to verify that the bits in $I_R \cap I_S \subset [n]$ will be decrypted correctly. Since I_R is uniformly chosen, and the bits in $I_R \cap I_S$ are decrypted correctly while the others will be in error with probability $\frac{1}{2}$, we can bound the probability of decryption error.

Lemma 1. The probability the receiver decrypts more than δn bits incorrectly is bounded by

$$\Pr[\text{ more than } \delta n \text{ bits are decrypted incorrectly}] \leq e^{-\frac{1}{2}c_r^2 c_s n} + e^{-2\left(\frac{c_r c_s}{2}+\delta-\frac{1}{2}\right)^2 n/\left(1-\frac{c_r c_s}{2}\right)}$$

Proof. First, we lower-bound $|I_R \cap I_S|$ (recall these bits are always decrypted correctly).

Since I_R and I_S are both chosen uniformly at random, we can imagine fixing I_R and choosing I_S at random. Now, let X_i be the indicator variable indicating whether the ith element of I_S is in I_R. Let $X = \sum_{i=1}^{|I_S|} X_i$. Thus $X = |I_R \cap I_S|$. Then $\Pr[X_i = 1] = c_r$, so $E(X) = c_r |I_S| = c_r c_s n$.

Since the X_i are negatively correlated we can apply a Chernoff bound ([12] Thms 1.1 and 4.3) to obtain

$$\Pr[X < E(X) - t] \leq e^{-2t^2/|I_S|}$$

taking $t = \frac{E(X)}{2} = \frac{c_r c_s n}{2}$ we have

$$\Pr\left[X < \frac{c_r c_s n}{2}\right] < e^{-\frac{1}{2}c_r^2 c_s n}$$

Now, for the elements in $[n] \setminus I_R \cap I_S$, these have a half chance of decrypting correctly. Let Y_i be the indicator random variable that indicates whether the ith element of $[n] \setminus I_R \cap I_S$ decrypts correctly. Thus $E(Y_i) = \frac{1}{2}$. If we let $Y = \sum_{i=1}^{n-|I_R \cap I_S|} Y_i$, we have $E(Y) = \frac{n-X}{2} = (n - |I_R \cap I_S|)/2$.

- **Key Generation:**
  Choose an error correcting code $\mathbf{ECC} : \{0,1\}^\ell \to \{0,1\}^n$ such that $n = \mathcal{O}(\ell)$ that is recoverable from a $\frac{1}{2} - \delta$ fraction of errors.[a] Sample $N \xleftarrow{\$} \mathsf{RSA}(\min(\lambda, \mathcal{O}(\ell \log \ell))$. Choose a strong $(\tau, 2^{-\lambda})$ extractor $\mathrm{Ext} : \{0,1\}^d \times \mathbb{Z}_N \to \{0,1\}$. Let $q_i = p_i^{e_i}$ where p_i is the $(i+2)nd$ prime and e_i is the smallest integer such that $q_i > 2^\tau$.

  Chooses a random subset $I_R \subset [n]$, with $|I_R| = c_r n$. Then

  - for $i \in I_R$, the receiver chooses $a_i \xleftarrow{\$} \mathbb{Z}_N$ and sets $h_i = \left(g^{\prod_{j \neq i} q_i}\right)^{a_i} \mod N$.
  - For $i \in [n] \setminus I_R$, The receiver chooses $h_i \xleftarrow{\$} \mathbb{Z}_N$.

  The receiver will know the decryption key, k_i, for the h_i with $i \in I_R$. The receiver will not be able to decrypt messages sent under public-keys h_i for $I \in [n] \setminus I_R$. As in the basic scheme, the factorization of N is *not* part of the secret key, and will not be used for decryption.

  $$\text{public key: } \{h_i\}_{i=1}^n \qquad\qquad \text{secret key: } \{a_i\}_{i \in I_R}$$

- **Encryption:**
  Given a message $m = m_1, \ldots, m_\ell \in \{0,1\}^\ell$, the sender encodes m as $(y_1, \ldots, y_n) = \mathbf{ECC}(m)$, then chooses a random $w \xleftarrow{\$} \mathbb{Z}_N$, and sets $x = g^w \mod N$. The sender chooses a random extractor seed, $\mathfrak{s} \xleftarrow{\$} \{0,1\}^d$. Then the sender chooses a random subset $I_S \subset [n]$, with $|I_S| = c_s n$ and

  - for $i \in I_S$, the sender sets

    $$c_i = \mathrm{Ext}_{\mathfrak{s}}\left((h_i)^w \mod N\right) \oplus y_i$$

  - for $i \in [n] \setminus I_S$, the sender chooses b_i uniformly from $\{0,1\}$, and sets $c_i = b_i$.

  The encryption randomness is $\{I_S, w, \mathfrak{s}, \{b_i\}_{i \in [n] \setminus I_S}\}$. The sender sends the ciphertext

  $$C = (x, \mathfrak{s}, c_1, \ldots, c_n) \in \mathbb{Z}_N \times \{0,1\}^d \times \{0,1\}^n$$

- **Decryption:**
  Given a ciphertext $C = (x, \mathfrak{s}, c_1, \ldots, c_n)$ and a secret key $\{k_i\}_{i \in I_R}$ the receiver decrypts as follows:

  - if $i \in I_R$ the receiver calculates

    $$y_i' = \mathrm{Ext}_{\mathfrak{s}}\left(\left(x^{\prod_{j \neq i} q_j}\right)^{a_i}\right) \oplus c_i = \mathrm{Ext}_{\mathfrak{s}}\left(\left((g^w)^{\prod_{j \neq i} q_j}\right)^{a_i}\right) \oplus c_i = \mathrm{Ext}_{\mathfrak{s}}\left(\left((g^{a_i})^{\prod_{j \neq i} q_j}\right)^{w}\right) \oplus c_i$$

  - if $I \in [n] \setminus I_R$ the receiver sets $y_i' = \perp$.

  Given $y_1', \ldots, y_n'$ the receiver decodes using the decoding algorithm for **ECC** to obtain a message $m_1', \ldots, m_\ell'$.

[a] Constant rate binary codes that can efficiently recover from a $\frac{1}{2} - \delta$ fraction of errors exist in the bounded channel model [19].

Figure 2: Non-committing encryption from Φ-hiding

Now X is the size of $|I_R \cap I_S|$ and Y is the number of bits outside of $I_R \cap I_S$ that are decrypted correctly. So the probability that more than δn of the bits are decrypted incorrectly is

$$
\begin{aligned}
&\Pr\left[X + Y < (1-\delta)n\right] \\
&\quad = \Pr\left[Y < (1-\delta)n - X\right] \\
&\quad = \Pr\left[Y < E(Y) - (E(Y) - (1-\delta)n + X)\right] \\
&\quad = \Pr\left[Y < E(Y) - \left(\frac{n-X}{2} - (1-\delta)n + X\right)\right] \\
&\quad = \Pr\left[Y < E(Y) - \left(\frac{X}{2} - \left(\frac{1}{2} - \delta\right)n\right)\right] \\
&\quad = \Pr\left[Y < E(Y) - \left(\frac{X}{2} - \left(\frac{1}{2} - \delta\right)n\right) \middle| X < \frac{c_r c_s n}{2}\right] \Pr\left[X < \frac{c_r c_s n}{2}\right] \\
&\quad + \Pr\left[Y < E(Y) - \left(\frac{X}{2} - \left(\frac{1}{2} - \delta\right)n\right) \middle| X \geq \frac{c_r c_s n}{2}\right] \Pr\left[X \geq \frac{c_r c_s n}{2}\right] \\
&\quad \leq e^{-\frac{1}{2}c_r^2 c_s n} + \Pr\left[Y < E(Y) - \left(\frac{X}{2} - \left(\frac{1}{2} - \delta\right)n\right) \middle| X \geq \frac{c_r c_s n}{2}\right] \\
&\quad \leq e^{-\frac{1}{2}c_r^2 c_s n} + \Pr\left[Y < E(Y) - \left(\frac{c_r c_s}{2} + \delta - \frac{1}{2}\right)n \middle| X \geq \frac{c_r c_s n}{2}\right] \\
&\quad \leq e^{-\frac{1}{2}c_r^2 c_s n} + e^{-2\left(\left(\frac{c_r c_s}{2} + \delta - \frac{1}{2}\right)n\right)^2 / \left(\left(1 - \frac{c_r c_s}{2}\right)n\right)} \quad \left(X \geq \tfrac{c_r c_s n}{2} \text{ and Chernoff}\right) \\
&\quad \leq e^{-\frac{1}{2}c_r^2 c_s n} + e^{-2\left(\frac{c_r c_s}{2} + \delta - \frac{1}{2}\right)^2 n / \left(1 - \frac{c_r c_s}{2}\right)}
\end{aligned}
$$

□

In the second to last line, recall that X is the number of summands in Y so the Chernoff bound for Y gets better as X increases.

Note that to apply Chernoff, we need $\frac{c_r c_s}{2} + \delta - \frac{1}{2} > 0$, in other words $\delta > \frac{1}{2} - \frac{c_r c_s}{2}$. Thus we will need a code **ECC** that can correct from an error-rate of $\frac{1}{2} - \frac{c_r c_s}{2}$. This can be achieved, and in fact constant-rate codes that can recover from a δ fraction of errors exist for any $\delta < \frac{1}{2}$ in the bounded channel model [19].

5 The Simulator

The simulation game in a non-committing encryption proceeds as follows: the simulator generates a public-key and a ciphertext. The adversary then chooses a target message, and the simulator must produce a secret key and encryption randomness such that the entire ensemble of public-key, secret key, ciphertext, encryption randomness and target message is indistinguishable from a valid run of the key-generation and encryption algorithms encrypting the target message.

- **Key Generation:** As in the real scheme the simulator makes use of an error correcting code **ECC** : $\{0,1\}^\ell \to \{0,1\}^n$, and an extractor $\mathrm{Ext} : \{0,1\}^d \times \mathbb{Z}_N \to \{0,1\}$.

  Note that unlike the valid sender and receiver, the simulator *will* need the factorization of the RSA modulus N (details below).

  - The simulator chooses a random extractor seed $\mathfrak{s} \xleftarrow{\$} \{0,1\}^d$.
  - The simulator uniformly chooses a subset $I_{\text{good}} \subset [n]$ of size $c_g n$, and sets $I_{\text{bad}} = [n] \setminus I_{\text{good}}$. Let $c_b = \frac{|I_{\text{bad}}|}{n} = 1 - c_g$.
  - The simulator generates an Φ-hiding modulus, N, such that N Φ-hides q_j for $j \in I_{\text{bad}}$.[6]

[6] This means that $N \geq \left(\prod_{j \in I_{\text{bad}}} q_j\right)^5$ to avoid Coppersmith's attack. Since $|I_{\text{bad}}| = \mathcal{O}(n)$, and the $p_i \approx i \log i$ by the Prime Number Theorem, we have $N \approx (n \log n)^{\mathcal{O}(n)}$, which implies $\log N = \mathcal{O}(n \log n)$.

– The simulator chooses $r \xleftarrow{\$} \mathbb{Z}_N$ and sets $g = r^{\prod_{j \in I_{\text{bad}}} q_j} \mod N$.

– The simulator then chooses keys $a_i \xleftarrow{\$} \mathbb{Z}_N$ for $i = 1, \ldots, n$.

– The simulator then outputs:

$$\text{public parameters: } N, g \qquad \text{Public key:} \{h_i\}_{i=1}^n$$

The public-key will be the collection $\{h_i\}_{i=1}^n$. A crucial point of our scheme is that the simulator knows the decryption keys for all h_i for all i, and will be able to decrypt for all $i \in I_{\text{good}}$, unlike in the real scheme, where the receiver only knows the keys for some smaller subset I_R.

- **Ciphertext Generation:**

  – The simulator generates $w \xleftarrow{\$} \mathbb{Z}_n$

  – The simulator partitions I_{good} into two random disjoint sets A_1 and A_0, with $|A_1| \approx |A_0| \approx c_g n/2$, and $A_1 \cup A_0 = I_{\text{good}}$.

  – For $i \in A_1$, the simulator sets $c_i = \text{Ext}_{\mathfrak{s}}(h_i^w \mod N) \oplus 1$ and for $i \in A_0$, the simulator sets $c_i = \text{Ext}_{\mathfrak{s}}(h_i^w \mod N)$.

  – For $i \in I_{\text{bad}}$ the simulator chooses a random bit b_i and sets $c_i = b_i$.

  The simulator outputs the ciphertext

$$C = (x, \mathfrak{s}, c_1, \ldots, c_n) \in \mathbb{Z}_N \times \{0,1\}^d \times \{0,1\}^n$$

- **Simulating Encryption Randomness:** After generating a public-key and a ciphertext, the simulator is given a target message m^*. The simulator calculates $y^* = \mathbf{ECC}(m^*)$.

  The receiver chooses the set I_S as follows: Let $M_1 = \{i : y_i^* = 1\}$, and $M_0 = \{i : y_i^* = 0\}$.

  If $|M_1 \cap A_1 \cup M_0 \cap A_0| < c_s n$ the simulator outputs $\perp$.

  We show in Section 5.1 that this happens with negligible probability.

  Otherwise, the simulator samples $c_s n$ elements uniformly from $M_1 \cap A_1 \cup M_0 \cap A_0$ and calls this set I_S.

- **Generating the Secret Key:** If $|M_1 \cap A_1 \cup M_0 \cap A_0| < c_r n$ the simulator outputs $\perp$.

  We show in Section 5.1 that this happens with negligible probability.

  Otherwise, the simulator samples $c_r n$ elements uniformly from $|M_1 \cap A_1 \cup M_0 \cap A_0|$ and calls this set I_R. Since $A_1 \cup A_0 = I_{\text{good}}$, and the receiver knows all the secret keys corresponding to the indices in I_{good} the simulator can output a_i for $i \in I_R$.

5.1 Good randomness and secret keys exists with overwhelming probability

The simulator outputs $\perp$ if $|M_1 \cap A_1 \cup M_0 \cap A_0| < \max(c_r, c_s)n$, thus we need to show that this happens with only negligible probability. Recall that M_i is the set of indices where the (encoded) target message has value i, whereas A_i is the set of indices where the ciphertext encrypts value i. The sets A_i were chosen uniformly at random, and the sets M_i were adversarially chosen (since they are determined by the adversary's choice of message).

We begin by showing that $|M_1 \cap A_1 \cup M_0 \cap A_0|$ will be large if M_0, M_1 were chosen independently of A_0, A_1. Then we will argue that the adversary cannot do much better than choosing M_0, M_1 independently.

If M_0 and M_1 were chosen independently of A_0, and A_1, then the probability that the simulator outputs $\perp$ can be bounded by Chernoff.

In particular, we have $E(|M_i \cap A_i|) = \frac{|A_i||M_i|}{n} = \frac{c_g|M_i|}{2}$ (since $|A_i| = \frac{c_g}{2}n$).

$$\Pr\left[|M_i \cap A_i| < \frac{c_g|M_i|}{2} - \frac{c_g|M_i|}{4}\right] < e^{-2\frac{c_g|M_i|^2}{8n}}$$

Since $|M_0| + |M_1| = n$, at least one of the $|M_i|$ will have $|M_i| \geq \frac{n}{2}$, thus

$$\Pr\left[|M_1 \cap A_1 \cup M_0 \cap A_0| < \frac{c_g n}{8}\right] < e^{-2\frac{c_g n}{32}}$$

which is negligible in n. Thus we have the desired result whenever $\max(c_r, c_s) < \frac{c_g}{8}$.

Now, the sets M_0, M_1 are not independent of A_0, A_1 because they were generated adversarially after seeing the ciphertext C (which encodes A_0, A_1). Since the adversary is polynomially-bounded, and the sets A_0 and A_1 are hidden by the semantic security of the underlying Φ-hiding based encryption, a standard argument shows that if the adversary could cause $|M_1 \cap A_1 \cup M_0 \cap A_0| > \min(c_r, c_s)n$ with non-neglible probability, then this adversary could be used to break the Φ hiding assumption. (Note that the simulator does *not* need the factorization of N to determine the size $|M_1 \cap A_1 \cup M_0 \cap A_0|$.)

5.2 Simulating the Choice of w

5.2.1 Simulating w

Most of the technical difficulty in this construction lies in finding a suitable w^* for the simulator to output. Since $g^w \mod N$ is part of the ciphertext, $g^{w^*} = g^w \mod N$.

Since g is a power of q_j for all $j \in I_{\text{bad}}$ this means $w = w^* \mod \varphi(N)/(\prod_{j \in I_{\text{bad}}} q_j)$. If we simply output $w^* = w$, the adversary can distinguish the simulator's output from a true output.

In a valid encryption for $i \in [n] \setminus I_s$ about half of the c_i should equal $\text{Ext}_{\mathfrak{s}}(h_i^w \mod N) + y_i$, but if the simulator simply outputs $w^* = w$, by using w^* to "decrypt" c_i for $i \in [n] \setminus I_S$ and the adversary will find that many fewer than half the locations $i \in [n] \setminus I_s$ match the encoded message. In the next section we will provide an algorithm for choosing w^*.

5.2.2 Finding a target set T

The simulator will have revealed secret keys for all $i \in I_R \subset M_1 \cap A_1 \cup M_0 \cap A_0 \subset I_{\text{good}}$. The simulator cannot lie about values whose indices are in I_{good}.

From the adversary's view, for $i \in [n] \setminus I_S$ the encryptions c_i will be encryptions of y^*. In an honest execution of the encryption algorithm, for $i \notin [n] \setminus I_S$, the encryptions, c_i should be uniformly random (in particular, they should be independent of y^*).

Thus the simulator needs to choose values in I_{bad} so that the c_i for $i \in [n] \setminus I_S$ encrypt values that are uniformly distributed on $\{0, 1\}$. This is because the honest encrypter will choose c_i at random for $i \notin I_S$.

Let $\alpha_b = \{i \in I_{\text{good}} \setminus I_S : \text{Ext}_{\mathfrak{s}}(g^w \mod N) \oplus b = c_i\}$. Thus α_b denotes the number of indices that are encryptions of b (that the simulator cannot change).

In an honest execution, the number of encryptions of 0 that an adversary sees in $[n] \setminus I_S$ is binomially distributed with parameters $n - |I_S| = (1 - c_s)n$ and $\frac{1}{2}$.

The simulator will sample a random variable, X, whose distribution is a binomial, shifted by α_1.

$$\Pr[X = j] = \binom{(1 - c_s)n}{j + \alpha_1} \left(\frac{1}{2}\right)^{(1-c_s)n} \quad \text{for } j = -\alpha_1, \ldots, (1 - c_s n) - \alpha_1$$

If $X > c_b n$ or $X < 0$ then the simulator will return $\perp$.[7]

Otherwise, the simulator will choose a random subset of I_{bad} of size X, and set those X elements 1, and the remaining $c_b - X$ elements to 0.

[7] Chernoff says that this happens with negligible probability. In particular, by Chernoff

$$\Pr[X < 0] < e^{-2\frac{\left(\frac{(1-c_s)n}{2} - \alpha_1\right)^2}{(1-c_s)n}}$$

and

$$\Pr[X < c_b n] < e^{-2\frac{\left(c_b - \frac{(1-c_s)n}{2}\right)^2}{(1-c_s)n}}$$

(see for example [5] Claim 4.6)

If we denote $I_{\text{bad}} = \{i_1, \ldots, i_{c_b n}\}$, then the simulator has chosen $\{t_{i_j}\}_{j=1}^{c_b n}$ so that exactly X of the t_{i_j} are 1, and $c_b n - X$ of the t_{i_j} are 0.

Now, consider the distribution of encryptions of 1 in the set $[n] \setminus I_S$. There are exactly $\alpha_1 + X$ ones in that set, and assuming the simulator does not output $\perp$, the probability of this occurring is exactly $\binom{(1-c_s)n}{\alpha_1+X} \left(\frac{1}{2}\right)^{(1-c_s)n}$ which is exactly the binomial distribution.

5.2.3 Calculating w^*

Denote $I_{\text{bad}} = \{i_1, \ldots, i_{c_b n}\}$. For $i \in I_{\text{bad}}$ the simulator has set $h_i = \left(g^{\prod_{j \neq i} q_j}\right)^{a_i} = \left(\left(r^{\prod_{j \in I_{\text{bad}}} q_j}\right)^{\prod_{j \neq i} q_j}\right)^{a_i} \mod N$. For the bad set I_{bad} the ciphertexts are "lossy", so the smoothness of the hash proof system makes the ciphertexts non-binding. Suppose the simulator has chosen target values $\{t_i\}_{i \in c_b n} \subset \{0,1\}^{c_b n}$ (using the methods in the previous section). The simulator's goal is for c_{i_j} to be an encryption of t_j for $j = 1, \ldots, c_b n$. We will show how the simulator can choose a w^* so that c_{i_j} looks like an encryption of t_j for $j = 1, \ldots, c_b n$. In particular, this means

$$\text{Ext}_{\mathfrak{s}}\left(h_{i_j}^{w^*} \mod N\right) = c_{i_j} \oplus t_j \quad \text{for } j = 1, \ldots, c_b n$$

Now, $w^* = w \mod \varphi(N)/\left(\prod_{j \in I_{\text{bad}}} q_j\right)$. The simulator then runs the following iterative algorithm (Algorithm 1) to find w^*. (Recall: $I_{\text{bad}} = \{i_1, \ldots, i_{c_b n}\}$.)

Algorithm 1 Calculating w^*

```
w* = w
for j = 1, ..., c_b n do
  while Ext_s(g^{w*} mod N) ≠ c_{i_j} ⊕ t_j do
    w* = w* + φ(N)/q_{i_j} mod φ(N)
  end while
end for
```

First, notice that at every step $w^* = w \mod \varphi(N)/\prod_{j \in I_{\text{bad}}} q_j$. Second, notice that when processing target index $i_j \in I_{\text{bad}}$, the value of w^* does change modulo q_{i_j}, but not for any other primes $q_k \in I_{\text{bad}} \setminus \{i_j\}$. Since g is *not* a q_{i_j}th power, the value g^{w^*} changes as w^* changes modulo q_{i_j}. Thus the expected running time of the while loop is only 2.

6 Efficiency and Parameters

A codeword in our cryptosystem is given by

$$C = (x, \mathfrak{s}, c_1, \ldots, c_n) \in \mathbb{Z}_N \times \{0,1\}^d \times \{0,1\}^n$$

Since $x \in \mathbb{Z}_N$, $\mathfrak{s} \in \{0,1\}^d$, and $c_i \in \{0,1\}$, the length of the codeword is $\log N + d + n$. Using the explicit extractors of [15], we can set the seed length d to be $\mathcal{O}(\log\log N + \log 1/\epsilon)$, so setting $\epsilon = 2^{-\lambda}$, we can choose $d = \mathcal{O}(\lambda)$, and extract bits that are negligibly close to uniform.

Now, the element $x \in \mathbb{Z}_N$, so $|x| = \log N$. Since the modulus, N, generated by the simulator must be able to Φ-hide all q_j for $j \in I_{\text{bad}} \subset [n]$. Thus $\sqrt[5]{N} \geq \prod_{j \in I_{\text{bad}}} q_j$ to avoid Coppersmith's attack. Since $|I_{\text{bad}}| = \mathcal{O}(n)$, and the $p_i \approx i \log i$ by the Prime Number Theorem so $N \approx (n \log n)^{\mathcal{O}(n)}$, which means that $\log N = \mathcal{O}(n \log n)$. This means the codewords are of length $\mathcal{O}(n \log n + \lambda + n) = \mathcal{O}(n \log n)$. In our scheme, the plaintext messages are of length ℓ, and $n = |\mathbf{ECC}(m)| = \mathcal{O}(\ell)$, so $\mathcal{O}(n \log n) = \mathcal{O}(\ell \log \ell)$. Thus our scheme achieves a logarithmic ciphertext expansion. By contrast, the best previously known schemes have a ciphertext expansion of λ [8]. By assumption, super-polynomial-time calculations in λ are infeasible while polynomial-time calculations in ℓ (and hence exponential-time calculations in $\log(\ell)$) are efficient. Thus $\lambda \gg \log \ell$, so our scheme achieves an improved ciphertext expansion.

λ	Security Parameter
ℓ	Message length
n	Codeword length ($n = \lvert\mathbf{ECC}(m)\rvert = \mathcal{O}(\ell)$)
N	Φ-hiding modulus ($\mathcal{O}(n \log n)$ bits)
δ	fraction of errors tolerated by **ECC** ($\delta > \frac{1}{2} - \frac{c_r c_s}{2}$, so $\delta = \mathcal{O}(1)$)
c_r	fraction of indices where receiver can decrypt ($c_r = \mathcal{O}(1)$)
c_s	fraction of indices correctly encoded by sender ($c_s = \mathcal{O}(1)$)
c_g	fraction of "good" indices known by the simulator ($\max(c_r, c_s) < \frac{c_g}{8}$)
d	Length of extractor seed ($d = \mathcal{O}(\lambda)$)
$\log N + d + n$	Codeword length
$n \log N$	Public-key length
$c_r n \log N$	Private-key length

Table 1: Summary of the parameters in our scheme

7 Conclusion

In this work, we constructed the first non-committing encryption scheme with messages of length $|m|$ and ciphertexts of length $\mathcal{O}(|m| \log |m|)$. This improves on the best prior constructions which had ciphertexts of length $\mathcal{O}(|m|\lambda)$ [8]. Like most previous protocols, a public key in our system consists of n public-keys for which the receiver has decryption keys for only a small subset. The sender then encodes the message using an error-correcting code, and encrypts each bit of the encoded message using a different key. Since each bit of the encoded message is encrypted as a separate ciphertext, a blowup of $\mathcal{O}(\lambda)$ seems unavoidable. The main technical contribution of our work is to show that by using the Φ-hiding assumption, we can encode each bit of the encoded message using a different key, but the same randomness, thus eliminating the factor $\mathcal{O}(\lambda)$ ciphertext blowup. Many cryptosystems allow randomness re-use across different keys, (indeed El-Gamal has this property, and this is the basis of most Lossy-trapdoor function constructions [20, 16]). The difficulty arises in generating a system where false randomness can be *efficiently* generated when the randomness is re-used across multiple keys. To achieve this, we turn to the Φ-hiding assumption [4].

It remains an interesting open question whether this protocol can be generalized, e.g. implemented with any smooth projective hash proof system, or whether the ciphertext blowup can be reduced to $\mathcal{O}(1)$. It is also open to construct a non-committing encryption scheme with n-bit messages and public-keys of size $\mathcal{O}(n + \lambda)$.

References

[1] Donald Beaver. Plug and Play Encryption. In *Crypto '97*, pages 75–89, London, UK, 1997. Springer-Verlag.

[2] Mihir Bellare, Alexandra Boldyreva, Kaoru Kurosawa, and Jessica Staddon. Multirecipient Encryption Schemes: How to Save on Bandwidth and Computation Without Sacrificing Security. *IEEE Transactions on Information Theory*, 53(11):3927–3943, November 2007.

[3] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *STOC*, STOC '88, pages 1–10, New York, NY, USA, 1988. ACM.

[4] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally Private Information Retrieval with Polylogarithmic Communication. In *Advances in Cryptology: EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 402–414. Springer Verlag, 1999.

[5] Ran Canetti, Uri Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 639–648, New York, NY, USA, 1996. ACM.

[6] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally Composable Two-Party and Multi-Party Secure Computation. In *STOC '02*, 2002.

[7] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty Unconditionally Secure Protocols. In *STOC*, pages 11–19, 1988.

[8] Seung G. Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Improved Non-committing Encryption with Applications to Adaptively Secure Protocols. In *ASIACRYPT '09*, 2009.

[9] Don Coppersmith. Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities. *Journal of Cryptology*, 10(4):233–260, 1997.

[10] Ronald Cramer and Victor Shoup. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In - *Eurocrypt 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64, 2002.

[11] Ivan Damgård and Jesper B. Nielsen. Improved Non-committing Encryption Schemes Based on a General Complexity Assumption. In *Crypto '00: Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, pages 432–450, London, UK, 2000. Springer-Verlag.

[12] Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, March 2012.

[13] Craig Gentry and Zulfikar Ramzan. Single-Database Private Information Retrieval with Constant Communication Rate. In *Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, pages 803–815. Springer Berlin / Heidelberg, 2005.

[14] Oded Goldreich, Sylvio Micali, and Avi Wigderson. How to Play any Mental Game. In *STOC '87*, pages 218–229, 1987.

[15] Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. Unbalanced Expanders and Randomness Extractors from Parvaresh–Vardy Codes. *J. ACM*, 56(4), July 2009.

[16] Brett Hemenway and Rafail Ostrovsky. Extended-DDH and Lossy Trapdoor Functions. In *PKC 2012*, 2012.

[17] Eike Kiltz, Adam O'Neill, and Adam Smith. Instantiability of RSA-OAEP under chosen-plaintext attack. In *Proceedings of the 30th annual conference on Advances in cryptology*, CRYPTO'10, pages 295–313, Berlin, Heidelberg, 2010. Springer-Verlag.

[18] Mark Lewko, Adam O'Neill, and Adam Smith. Regularity of Lossy RSA on Subdomains and Its Applications. In Thomas Johansson and PhongQ Nguyen, editors, *Advances in Cryptology EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 55–75. Springer Berlin Heidelberg, 2013.

[19] Silvio Micali, Chris Peikert, Madhu Sudan, and David A. Wilson. Optimal Error Correction Against Computationally Bounded Noise. In *TCC '05*, pages 1–16, 2005.

[20] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *STOC '08: Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 187–196, New York, NY, USA, 2008. ACM.