Sponge based CCA2 secure asymmetric encryption for arbitrary length message

Tarun Kumar Bansal Donghoon Chang
Somitra Sanadhya
Indraprastha Institute of Information Technology, Delhi (IIIT-D), India
{tarunb,donghoon,somitra}@iiitd.ac.in

Abstract

OAEP and other similar schemes proven secure in Random-Oracle Model require one or more hash functions with output size larger than those of standard hash functions. In this paper, we show that by utilizing popular Sponge constructions in OAEP framework, we can eliminate the need of such hash functions. We provide a new scheme in OAEP framework based on Sponge construction and call our scheme *Sponge based asymmetric encryption padding* (SpAEP). SpAEP is based on 2 functions: Sponge and SpongeWrap, and requires only standard output sizes proposed and standardized for Sponge functions. Our scheme is CCA2 secure for any trapdoor one-way permutation in the ideal permutation model for arbitrary length messages. Our scheme utilizes the versatile Sponge function to enhance the capability and efficiency of the OAEP framework. We also propose a key encapsulation mechanism and a tag-based key encapsulation mechanism for hybrid encryption using SpAEP with any trapdoor one-way permutation. Our scheme SpAEP utilizes the permutation model efficiently in the setting of public key encryption in a novel manner.

Keywords: OAEP, Sponge function, public key encryption, hybrid encryption, CCA2 security.

1 Introduction

The Optimal Asymmetric Encryption Padding (OAEP), proposed by Bellare and Rogaway at Eurocrypt '94 [1], is a technique for converting the RSA trapdoor permutation into a chosen ciphertext secure system in the random oracle model (ROM). In Crypto '01, Shoup described a modification to OAEP, called OAEP+, that provably converts any trapdoor one way permutation (f) into a chosen ciphertext secure system in the random oracle model. In 2003, Phan and Pointcheval [2, 3] introduced OAEP-3R which is RCCA secure ("relaxed CCA" [3] equivalent to "replayable CCA" [4]- a slightly weaker notion than general CCA2) with any trapdoor one way permutation (f) in ROM. Let "ciphertext overhead" [5] stand for the difference between the length of ciphertext and plaintext. OAEP-3R was shown to have only t-bit ciphertext overhead, whereas OAEP and OAEP+ have $3t$-bit ciphertext overhead, where t stands for security requirement in bits[1]. In 2008, Abe, Kiltz and Okamoto [5] showed that security reduction of OAEP-3R forces ciphertext overhead to be $2t$. A new scheme called OAEP-4X was introduced in [5] which provides CCA2 security for any trapdoor one way permutation in ROM. OAEP-4X has only t-bit ciphertext overhead which was shown to be optimal (lowest achievable bound). In OAEP-4X, reduction of t-bit ciphertext overhead with respect to OAEP-3R has only limited practical application such as in a highly bandwidth

[1]A security requirement of t-bit implies that at-least 2^t queries are required to break the scheme with probability close to 1

constrained network. Therefore, for general applications ciphertext overhead reduction by t-bits is a less interesting case.

Number of hash functions used in OAEP is 2 and these are used in a 2 round structure. OAEP+ is also 2 round structure but uses 3 hash functions (2 hash function can run in parallel while encryption). OAEP-3R is 3 round structure that uses 3 hash functions and OAEP-4X is 4 round structure that uses 4 hash functions. Each of these schemes (OAEP, OAEP+, OAEP-3R and OAEP-4X), proven secure in ROM, requires one or more hash functions with arbitrary size output. For example, for RSA-2048 (or RSA-3072) trapdoor one-way permutation, minimum number of hash function with arbitrary size output required in OAEP, OAEP+, OAEP-3R and OAEP-4X are 1, 1, 2 and 2 respectively.

Currently, no cryptographic standard specifies an instantiation for hash function of arbitrary size. However, some instantiations are implicit in PKCS #1 v2.1 [6], because RSA-OAEP [1] are standardized. For example, RSA-OAEP requires two random hash functions G and H with small input size (less than the RSA modulus) and arbitrary size output, which are both instantiated in PKCS by the MGF1 pseudo-random number generator [6]. MGF1 uses a hash function in counter mode: $\mathrm{MGF1}(x) = h(x\langle count0\rangle)||h(x\langle count1\rangle)||h(x\langle count2\rangle)||\ldots$, where h is either SHA-1 or a SHA-2. Because MGF1 is not a regular standardized hash function, we use a term "non-standard hash function" for such functions, which instantiate a hash function of arbitrary output size by utilizing standard fixed length hash functions (e.g., SHA-1, SHA-2) to generate arbitrary hash output. Similarly, in other OAEP-type schemes, instantiation of such hash functions is done by using similar "non-standard hash function".

OAEP-type schemes (OAEP, OAEP+,OAEP-3R) discussed above, work only for restricted message length (less than input size of trapdoor one-way permutation) except OAEP-4X, which works for long messages (more than input size of trapdoor one-way permutation) as well. To encrypt lengthy messages, OAEP-4X uses one extra hash function and a passively secure symmetric encryption scheme along with 4 hash functions. In OAEP-4X, the ability of handling long messages is the result of utilizing well known Tag-KEM/DEM framework [7, 8]. Tag-KEM/DEM is considered a hybrid encryption scheme [9, 7, 10, 11, 12, 13, 14, 15]. In hybrid paradigm, an asymmetric key encapsulation mechanism (KEM) combines with a symmetric data encapsulation mechanism (DEM). Traditionally, KEM is a probabilistic algorithm that produces a random symmetric key and an asymmetric encryption of that key as the key encapsulation. DEM is a deterministic algorithm that takes a symmetric key, generated by KEM, and encrypts the message under that key. In Tag-KEM/DEM framework, KEM takes a feedback, referred to as the 'tag', from DEM part and then generates key encapsulation. Final ciphertext results from concatenation of key encapsulation and encryption of message. This traditional hybrid paradigm suffers from high ciphertext overhead (difference between plaintext and ciphertext length) equivalent to the size of asymmetric encryption of key.

In 2007, Bjørstad et.al. [8] introduced KEMs (RKEM and Tag-RKEM) with partial message input/recovery. These KEMs help in significant reduction of ciphertext overhead in hybrid constructions. [8] also showed that the Tag-RKEM is more space efficient than the RKEM in terms of the ciphertext overhead. For construction of RKEMs, [8] focuses over those asymmetric encryption schemes which can recover the random variable used in encryption, during the decryption, like in OAEP-type schemes. Therefore, [8] provided the use of RSA-OAEP in RKEMs as an example. OAEP-4X has also utilized the same idea proposed in [8] along with some improvement in Tag-KEM part. This signifies that the OAEP-type schemes are good candidates for constructing RKEMs and successive improvements which took place in OAEP-type schemes helped in the instantiation of the RKEMs also.

Motivation: Almost all previous public key cryptography literature dealing with OAEP-based encryption, requires a perfect random function (a Random oracle) over an arbitrary domain, whereas in practice one is given a random function or permutation over a relatively small domain: practical block-cipher, hash functions and permutations have smaller block size and fixed output length. Therefore, for generating lengthy hash output, RSA-Full Domain Hash [16, 17, 18] or the Mask Generation Function (MGF1) [6] in RSA-OAEP are currently implemented with a complex construction of fixed length hashes and counters. For m blocks input and n blocks hash output, a fixed length hash function has to run approximately $m \times n$ times. All of above mentioned schemes (OAEP, OAEP+, OAEP-3R, OAEP-4X) proven secure in ROM require one or more hash functions with output size larger than standard sizes (e.g., SHA-1, SHA-512). [19] showed that the hash function instantiation proposed in the literature for such cases are weaker than a random oracle, where hash functions are assumed to behave like random oracle in the security analysis. The “non-standard hash function” (like MGF1) are not well analyzed in literature, have complex construction of fixed length hash functions with counter and are also proven weaker than random oracle. This raises a question on the possibility of modifying the OAEP framework which does not require any “non-standard hash function” and where all the computations are performed in standardized input-output settings.

Development of schemes from OAEP to OAEP-4X shows differences in the number of rounds, depending upon calls to the hash functions used. OAEP and OAEP+ is considered as 2 round structure, OAEP-3R as 3 round and OAEP-4X as 4 round. This naturally poses a question on the possibility of further development of the OAEP-type scheme and reduce the number of rounds.

We have seen OAEP-type construction are good candidate in hybrid encryption for constructing KEMs like in [8] and improvements in OAEP-type construction helps the RKEMs and hybrid encryption also.

Presently, major existing and proposed crypto-systems are based on standard assumptions or proven secure in random oracle model for public key cryptography. The crypto-systems based on permutations, proven secure in ideal permutation model, and using them efficiently are yet to be explored to develop new outlooks and techniques that can cross-pollinate and advance cryptography as a whole. An open problem mentioned in [7] about having a hybrid construction from different primitives like permutation, also helps us to pursue in this direction.

Interestingly, popular Sponge constructions [20], based on iterative permutation, are found to be a suitable solution of all the questions mentioned above. Sponge constructions work in standardized input-output settings [21, 20, 22, 23] and are useful for encryption, Authentication Encryption (AE), variable length input/output and for MAC generations [24]. In a Sponge function, for m blocks input and n blocks hash output, roughly $m + n$ calls of internal primitive permutation are required. Other than that, number of permutation calls in a Sponge function [20], used as hash function, and SpongeWrap [22], a modification of Sponge function used as AE, is equal in general. Therefore, versatility of Sponge function encourages the designers to come up with more useful and efficient design. The glimpse of popularity of Sponge functions can be seen clearly in CAESAR [25] and PHC [26] competition. Therefore, it is interesting to consider a permutation based or more concretely a Sponge construction based OAEP-type scheme having a security proof in ideal permutation model which can also be utilized in hybrid encryption.

Our Contribution: In this work, we introduce a *Sponge based Asymmetric Encryption Padding* scheme (SpAEP), a novel way to use the SpongeWrap [22] and the Sponge function [20] to encrypt arbitrary length messages in Asymmetric key cryptography. Each function (SpongeWrap and Sponge) iterates a public invertible permutation as a primitive function. Permutation calls in

Sponge function and in a SpongeWrap are generally the same for equal number of input-output data blocks.

- We provide new direction for constructing asymmetric key cryptographic schemes in ideal permutation model by utilizing permutations, having smaller/practical domain, in SpAEP. Almost all previous public key cryptography literature dealing with OAEP-based encryption requires hash functions (or a random function) over an arbitrary domain.
- SpAEP uses the Sponge function and the SpongeWrap in standard input-output settings, proposed for "Sponge functions" [21, 20, 22, 23], as per security requirement. Therefore, SpAEP remove the requirement of having "non-standard hash function", which is required in OAEP, OAEP+, OAEP-3R and OAEP-4X for generating hash output of different sizes than standard size ("non-standard output size").
- In SpAEP, both functions (Sponge function and SpongeWrap) are used in parallel during encryption to speed-up the process. Therefore, we consider SpAEP as 1 round structure in comparison to other OAEP-type schemes. However, the functions are not parallelized during decryption.
- The construction SpAEP with arbitrary trapdoor one-way permutation also suggests two variant of Key encapsulation method (KEM), one is SpRKEM and second is Tag-SpRKEM. Conceptually, our approach is similar to the scheme *Tag-KEM with partial ciphertext recovery* [8] but in our case the message can be directly recovered. Therefore, Tag-SpRKEM can be used as a *Tag-KEM with partial message recovery.* This Tag-SpRKEM version is similar to OAEP-4X, which trivially take us to the comparison among OAEP-4X and SpAEP and other OAEP-type schemes.

Features of SpAEP and Comparison with other OAEP-type schemes

- Although the permutation used in Sponge is invertible, we do not use this fact for our construction and provide inverse-freeness. Therefore our construction allows using permutations which are inefficient to invert but efficient in the forward direction. That is, computation time, implementation or memory efficiency of the forward direction of the permutation can be exploited by user in our design. Moreover, our design allows using a non-invertible mapping in the Sponge function.
- Let f be a trapdoor one-way permutation then we denote the instantiation of our scheme with f by f-SpAEP. The f-SpAEP can process arbitrary length messages. Our scheme is CCA2 secure when used with any trapdoor one-way permutation.
- We provide a formal security proof of f-SpAEP in adaptive chosen ciphertext attack (CCA-2) notion in the ideal permutation model. Instead of directly using security proof, in ROM, of sponge construction, we prefer a dedicated proof from scratch in ideal permutation model to avoid multi-stage game problem [27, 28]. Although [2] introduced an efficient scheme in ideal permutation model with full domain permutation encryption, still it remained impractical because having such big permutation size equivalent to trapdoor one-way permutation size is itself hard. Similar problem of output size occurs when a scheme requires hash output different (generally larger) than standard sizes.
  In Table 1, we compare OAEP [1], OAEP+ [29], OAEP-3R [2] and OAEP-4X [5] with SpAEP. In Table 1 cipher text overhead values are taken from Table 1 [5].
  OAEP, OAEP+ and OAEP-3R can only handle messages of length less than input size of trapdoor one-way permutation unlike OAEP-4X and SpAEP can handle any message size.

	OAEP [1]	OAEP+ [29]	OAEP-3R [2]	OAEP-4X [5]	SpAEP
Ciphertext-overhead	$3t$	$3t$	$2t$	t	$2t$
# Function calls	2 Hash	3 Hash	3 Hash	5 Hash, 1 Symmetric Encryption (E)	1 SpongeWrap, 1 Sponge function
# Function calls Sequential or Parallel (Encryption)	Sequential	2 parallel, 1 sequential	Sequential	Mixed	Parallel
Trapdoor Perm.-f	RSA, Rabin	Any f	Any-f	Any f	Any-f
Max. Message size with f	$\ell - 3t$	$\ell - 3t$	$\ell - 2t$	Any	Any

Table 1: Comparison of OAEP, OAEP+, OAEP-3R, SpAEP, OAEP-4X Here t is the security requirements in term of number of bits. In order to break the scheme with probability 1 number of queries required are 2^t. ℓ is input-output size of trapdoor one way permutation f.

In Table 1, for OAEP, number of functions run in parallel during encryption is zero, both hash function run sequentially. Similarly in the case of OAEP-3R, all three hash function calls are sequential. OAEP+ uses three hash functions, out of which only two function calls can run in parallel. In OAEP-4X, for messages having size less than input size of trapdoor one-way permutation, only 4 hash function calls are required sequentially. For long messages(message size more than input size of trapdoor one-way permutation), OAEP-4X uses 5 hash function (H1, H2, H3, H4, G) calls and one symmetric encryption scheme (E). Initially only two hash function calls run in parallel (H1,G) then H2 and E runs parallel, and then H3 and H4 runs sequentially. Overall in OAEP-4X, for long messages, only two functions calls can run in parallel at instant. From Table 1, we can clearly see that SpAEP is a simple and efficient scheme in comparison to other schemes. Although SpAEP has t-bit extra ciphertext overhead with respect to OAEP-4X, yet as explained earlier this is a minor concern in general applications.

One may argue that in OAEP based schemes, for final encryption and decryption, trapdoor one-way permutation show dominance in computation time with respect to OAEP type structure. This makes each and every scheme in-advantageous over other schemes in terms of computation efficiency. We hope this may not be the case always or in future. Recent development in lattice based cryptography [30, 31, 32] shows up that computation time of trapdoor function can reduce significantly compared to existing traditional trapdoor permutation. Therefore, it is fruitful to have OAEP type schemes that are better than existing one and compatible with latest trends.

In summary, we are proposing an asymmetric padding scheme which is more simple and efficient in terms of structure and functionality than other existing OAEP-type schemes.

2 Preliminaries

We discuss some preliminaries in this section.

Ideal Permutation: An permutation π is a bijective function on a finite domain D and finite range R with $D = R$. An ideal permutation is a permutation chosen uniformly at random from all the available permutations. Let $D = R = \{0,1\}^b$, then $\pi \xleftarrow{\$} \text{Perm}(D, D)$, where $\text{Perm}(D, D)$ is the collection of all permutations on D. Mathematically, $\pi : D \rightarrow R$ is a permutation, if for every $y \in R$ there is one and only one $x \in D$ such that $\pi(x) = y$.

Trapdoor one way Permutation and their security: We recall the security notion of a trapdoor one way permutation scheme. This scheme requires a *trapdoor permutation generator*. This is a PPT algorithm $\mathcal{F}$ such that $\mathcal{F}(1^\ell)$ outputs a pair of deterministic algorithms (f, f^{-1}) specifying a permutation and its inverse on $\{0,1\}^\ell$. We associate to $\mathcal{F}$ an *evaluation time* $t_{\mathcal{F}}(\cdot)$: for all l, all $(f, f^{-1}) \in [\mathcal{F}(1^\ell)]$ and all $C \in \{0,1\}^\ell$, the time to compute $f(C)$ (given f and C) is $t_{\mathcal{F}}(\ell)$. Note that the evaluation time depends on the setting: for example on whether or not there is hardware available to compute f.

We will be interested in two attributes of a (possibly non-uniform) algorithm $\mathcal{B}$ trying to invert $\mathcal{F}(1^\ell)$-distributed permutations; namely its running time and its success probability.

Definition 1. *Let $\mathcal{F}$ be a trapdoor permutation generator. We say that f is a one-way trapdoor permutation if for any efficient adversary $\mathcal{B}$*

$$Pr[(f, f^{-1}) \leftarrow \mathcal{F}(1^l); y \xleftarrow{\$} \{0,1\}^l : \mathcal{B}(f, y) = C \text{ such that } f(C) = y] \leq \epsilon,$$

where ϵ is negligible in ℓ.

CCA, CCA2 Security: For probabilistic asymmetric key encryption algorithm, indistinguishability is defined by the following experiment between an adversary $\mathcal{A}$ and a challenger. For schemes based on computational security, the adversary is modeled by a probabilistic polynomial time Turing machine, meaning that it must complete the game and output a guess within a polynomial number of time steps. The scheme PE comprise $(\mathcal{G}, \mathcal{E}_{pk}, \mathcal{D}_{sk})$. Secret/Private keys ($pk$, sk) is the generated by $\mathcal{G}$ and s is the auxiliary information collected by $\mathcal{A}$ using oracle $\mathcal{O}_1$. $\mathcal{E}_{pk}(M)$ represents the encryption of a message M under the public key pk and $\mathcal{D}_{sk}$ represents the decryption of ciphertext y under secret key sk.

```
Experiment: $Exp_{PE,\mathcal{A}}^{ind-atk-d}(\ell)$
1. (pk,sk)$\xleftarrow{\$} \mathcal{G}(1^\ell)$; $(M_0, M_1, s) \leftarrow \mathcal{A}^{\mathcal{O}_1(\cdot)}$; $y \leftarrow \mathcal{E}_{pk}(M_d)$;
2. $d' \leftarrow \mathcal{A}^{\mathcal{O}_2(\cdot)}(M_0, M_1, y, s)$;
3. return $d'$;
```

where $|M_0| = |M_1|$ and y cannot be queried to $\mathcal{O}_2$ oracle, and

atk=CCA-1 :	$\mathcal{O}_1(\cdot) = D_{sk}(\cdot)$ and $\mathcal{O}_2(\cdot) = \mathcal{E}_{pk}$
atk=CCA-2 :	$\mathcal{O}_1(\cdot) = D_{sk}(\cdot)$ and $\mathcal{O}_2(\cdot) = D_{sk}(\cdot)$

The scheme is IND-CCA1/IND-CCA2 [33] secure if adversary $\mathcal{A}$ has a non-negligible advantage in winning ($d' = d$) the above game. The definition of security we have presented here is from [34]. It is known to be equivalent to other notions, such as non-malleability [33, 35] , which is called *NM-CCA-2* in [33].

$$|\Pr[Exp^{ind-cca2-d}_{PE,\mathcal{A}}(\ell) = d \mid d \xleftarrow{\$} \{0,1\}]| \leq negl(\ell) + \tfrac{1}{2}.$$

3 General View of OAEP+

In this section we provide a general view[2] of the OAEP+ with f as the trapdoor one way permutation in an informal way. This helps us to elaborate the basis of the design of our scheme. This general view is shown in Figure 2, 3. It has three parts:

1. *One time Authenticated Encryption (OAE)*: This is a one time authentication encryption that uses a one time key R and generates an encoded message C and Tag T_1 of message M. Message will be padded to suitable length according to OAE.
2. *Hash*: This is a deterministic hashing algorithm. The concatenation of the outputs of OAE with a one time key R is the input of this hashing algorithm. It outputs T_2.
3. *Trapdoor one way Permutation*: This is a trapdoor one way permutation $f : \{0,1\}^\ell \to \{0,1\}^\ell$, which takes the concatenation of the outputs of OAE and *Hash* and produces the final encryption.

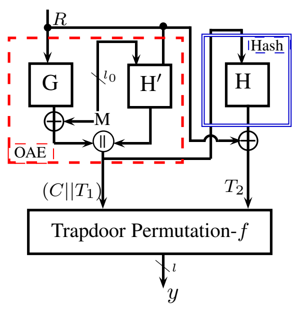


Figure 1: OAEP+ with f

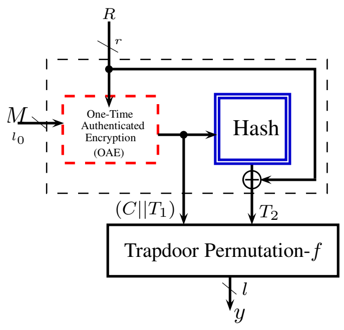


Figure 2: General View of Fig. 1

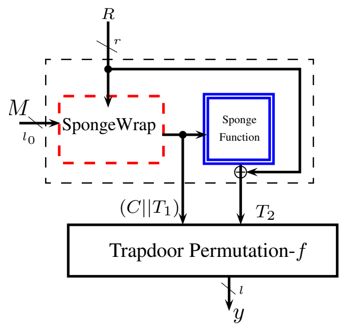


Figure 3: Sponge based view of Fig. 2

Figure 1 shows OAEP+ construction with f as the trapdoor one way permutation. G, H' and H are the hash functions used in OAEP+. If we map OAEP+ on our general view then the combination of G and H' is OAE while H is the Hash part. G provides a kind of one time pad encryption (OTE) to message M, H' provides hash tag T_1 of M and H produces hash tag T_2 of OTE and tag T_1.

In this work, we provide f-SpAEP as an example of this general view where the f-SpAEP scheme uses SpongeWrap as *OAE* and a Sponge function as *Hash* part with different IV.

One can view and use our scheme in hybrid encryption model and can also utilize in key/data encapsulation method. But in this paper we are more focused over sponge applicability in OAEP framework.

[2]This informal general view helps in understanding our scheme.

4 SpAEP-Sponge based Asymmetric Encryption Padding

SpAEP is a sponge function based construction. SpAEP iterates a fixed permutation $\pi : \{0,1\}^{b_i} \times \{0,1\}^{b_c} \rightarrow \{0,1\}^{b_i} \times \{0,1\}^{b_c}$ in a exact way to the Sponge construction and SpongeWrap [22, 20, 21].

The bit length of input and output of π, called bit rate, is $b = b_i + b_c$. The term b_i is called input rate and the term b_c is called capacity rate. The permutation π is the only underlying cryptographic primitive used by SpAEP. For using SpAEP for asymmetric key setting, one can use any trapdoor one way permutation $f : \{0,1\}^{\ell} \rightarrow \{0,1\}^{\ell}$ such as RSA. The resulting scheme is called f-SpAEP. The output of function f is $y \in \{0,1\}^{\ell}$ and the trapdoor of f is represented by f^{-1}. $\lfloor x \rfloor_r$ represents the first r bit of x or we can say it r-bit chop function. Similarly, $\lceil x \rceil$ represents the last r bit of x. See Figure 4 for graphical presentation.

SpAEP handles the arbitrary length message and π is fixed length permutation of input rate b_i. SpAEP uses a reversible padding function $pad(\cdot)$ to generate b_i-bit length blocks. For scheme to be compatible with f, scheme uses $pad(\cdot)$ to divided the message into two parts(M^n, M^e). First part should have minimum n-block message where each block is of b_i-bit length such that $\ell = n * b_i + 2r$, $\ell \geq b_i$ and $n \geq 1$. SpAEP process this n-block message $M^n = m_1 || \ldots || m_n$ and gives output $c_1 || \ldots || c_n$ which is the input of f. SpAEP process the second part of the message $M^e = m_{n+1} || \ldots || m_e$ and gives output $C_e = c_{n+1} || \ldots || c_e$. If M^e is an empty string, then C^e will also remain as empty string. SpAEP also outputs r-bit tag T_1 and T_2 which is also the input of f. Final output of the f-SpAEP will be $y || C_e$, where $y = f(C^f = c_1 || \ldots || c_n || T_1 || T_2)$. In case (ℓ, b_i, r) are chosen such that n is not an integer value, then $\lceil c_n \rceil_{|C^f| - \ell}$ bits removes and append to C^e to ensure $|C^f| = \ell$. Then $C^f = c_1 || \ldots || \lfloor c_n \rfloor_{(b_i - (|c_{[1,n]}| - \ell))}$ and $C^e = \lceil c_n \rceil_{(|c_{[1,n]}| - \ell)} || c_{n+1} || \ldots || c_e$. In paper we will consider n as integer only which means $|C^f| = \ell$. Accordingly, Encryption and Decryption of f-SpAEP are described in Algorithms 1 and 2 respectively.

The encryption and decryption procedures of SpAEP use forward direction of the permutation. Therefore we can have a permutation that is more efficient in forward direction than in inverse.

CCA2 Security of f-SpAEP: Next we provide a formal proof of CCA2 security of f-SpAEP. As described in Section 2, the experiment of adversary $\mathcal{A}$ for f-SpAEP is the following.

Experiment: $Exp^{ind-cca2-d}_{f-SpAEP,\mathcal{A}}(\ell)$

- $(\underbrace{f}_{\textbf{pk}}, \underbrace{f^{-1}}_{\textbf{sk}}) \overset{\$}{\leftarrow} \mathcal{F}(1^{\ell})$;
- $(M_0, M_1, s) \leftarrow \mathcal{A}^{\pi(\cdot), D_{f^{-1}}(\cdot), f}$;
- $y^* \leftarrow \mathcal{E}_f(M_d)$; $\ldots$ **one time encryption query**
- $d' \leftarrow \mathcal{A}^{\pi(\cdot), D_{f^{-1}}(\cdot), f}(M_0, M_1, y^*, s)$;
- **return** d';

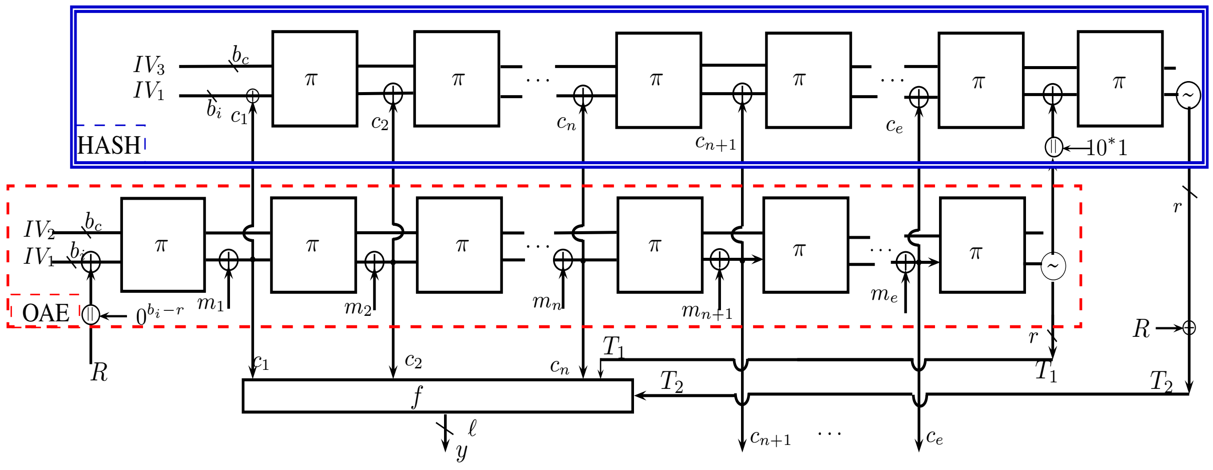


Figure 4: SpAEP with any trapdoor one way permutation f and public invertible permutation $\pi : \{0,1\}^{b_i} \times \{0,1\}^{b_c} \leftarrow \{0,1\}^{b_i} \times \{0,1\}^{b_c}$. SpAEP accepts message M and internally OAE call pad(M)=$m_1||\ldots||m_n||m_{n+1}||\ldots||m_e$ such that $n = (\ell - 2r)/b_i$, $|pad(M)| \geq (\ell - 2r)$ and each message bock is of length b_i where ℓ is size of trapdoor permutation-f and $|R| = |T_1| = |T_2| = r$. The symbol $\circledtilde$ represents taking r-bit output from b_i-bit input. The symbol $\oplus\!\!\!\!\bigcirc$ represent concatenation.

```
Algorithm 1: Encryption: $\mathcal{E}_f(M)$
$\mathcal{E}_f(M)$=$f$-SpAEP$^\pi(M)$
1  Initialization:
   $IV_1 = 0^{b_i}$,$IV_2 = 0^{b_c}$,$IV_3 = IV_2 \oplus 1$,
   $w = IV_2, x = IV_1$
2  Random Nonce: $R \xleftarrow{\$} \{0,1\}^r$
3  $pad(M) = m_1||m_2||\ldots||m_e$,
   where $|m_i| = b_i \ \forall 1 \leq i \leq e$
4  $x = x \oplus R||0^{b_r - r}$
5  for $i = 1 \rightarrow e$ do
6      $(x||w) = \pi(x||w)$
7      $x = x \oplus m_i$
8      $c_i = x$
9  $(x||w) = \pi(x||w)$; $T_1 = \lfloor x \rfloor_r$
10 $x = IV_1$ and $w = IV_3$
11 for $i = 1 \rightarrow e$ do
12     $x = x \oplus c_i$
13     $(x||w) = \pi(x||w)$
14 $x = x \oplus T_1||0^{b_r - r}$
15 $(x||w) = \pi(x||w)$
16 $T_2 = \lfloor x \rfloor_r \oplus R$
17 $C^f = c_1||c_2||\ldots||c_n||T_1||T_2$;
   $C^e = c_{n+1}||\ldots||c_e$
18 $y = f(C^f)$
19 Return: $y||C^e$
```

```
Algorithm 2: Decryption: $\mathcal{D}_{f^{-1}}(y||C^e)$
$\mathcal{D}_{f^{-1}}(y||C^e)$= $f$-SpAEP$^\pi(y||C^e)$, $f^{-1}$
1  Initialization: $IV_1 = 0^{b_i}$,$IV_2 = 0^{b_c}$,$IV_3 = IV_2 \oplus 1$,
   $w = IV_3, x = IV_1$
2  $C^f = c_1||c_2||\ldots||c_n||T_1||T_2 = f^{-1}(y)$;
   $c_{n+1}||\ldots||c_e = C^e$
3  $C' = c_1||c_2||\ldots||c_n||T_1||T_2||c_{n+1}||\ldots||c_e$, where
   $|c_i| = b_i, |T_1| = |T_2| = r$ for $1 \leq i \leq e$
4  for $i = 1 \rightarrow e$ do
5      $x = x \oplus c_i$
6      $(x||w) = \pi(x||w)$
7  $x = x \oplus T_1||0^{b_r - r}$
8  $(x||w) = \pi(x||w)$; $R = \lfloor x \rfloor_r \oplus T_2$
9  $x = R||0^{b_r - r}$; $w = IV_2$
10 for $i = 1 \rightarrow e$ do
11     $(x||w) = \pi(x||w)$
12     $m_i = x \oplus c_i$
13     $x = c_i$
14 $(x||w) = \pi(x||w)$; $T_1' = \lfloor x \rfloor_r$
15 if $T_1 = T_1'$ then
16     if $\exists\ M\ s.t.\ M = unpad(m_1||\ldots||m_e)$ then
17         Return:$M$
18     else
19         Return: Invalid
20 else
21     Invalid.
```

Theorem 1. *If the underlying trapdoor permutation f is one way, then f-SpAEP is secure against adaptive chosen cipher-text attack in the Ideal permutation model. The success probability of adversary $\mathcal{A}$ for CCA2 attack is*

$$Pr[Exp^{ind-cca2-d}_{f-SpAEP,\mathcal{A}}(\ell) = d | d \xleftarrow{\$} \{0,1\}] \leq \frac{1}{2} + \frac{(q-1)q}{2^{b+1}} + \frac{q(q+1)}{2^{b_c}} + \frac{5q_D}{2^r} + \frac{q_{\pi_\mathcal{A}} + q_{\pi^{-1}}}{2^r} + Adv^{owtp}_f(\mathcal{B}_\mathcal{A}\ Succeeds) + \frac{(q_{\pi_\mathcal{A}} + q_{\pi^{-1}})}{min(2^r, 2^{b_c})},$$

where q is the total number of (π and π^{-1}) queries, q_π and $q_{\pi^{-1}}$ are the number of π and π^{-1} queries, $q_{\pi_\mathcal{A}}$ is the number of π queries by $\mathcal{A}$, q_D is the number of decryption queries and b, b_c, r are the same as defined earlier, $\mathcal{B}$ is an adversary that finds the complete input C^f of trapdoor one way permutation f given $y \xleftarrow{\$} \{0,1\}^\ell$ such that $y = f(C^f)$, without having knowledge of f^{-1}. Adversary $\mathcal{B}$ uses $\mathcal{A}$ as a subroutine internally. $Adv^{owtp}_f(\mathcal{B}_\mathcal{A}\ Succeeds)$ is the success advantage that a particular adversary $\mathcal{B}$ has in breaking the trapdoor one-way permutation f. The time and space requirements of $\mathcal{B}$ are related to $\mathcal{A}$ as follows:

$$Time(\mathcal{B}) = O(Time(\mathcal{A}) + (q_{\pi_\mathcal{A}} + q_{\pi^{-1}}) \cdot t_f + (q_{\pi_\mathcal{A}} + q_{\pi^{-1}} + q_D) \cdot \ell); \tag{1}$$

$$Space(\mathcal{B}) = O(Space(\mathcal{A}) + (q_{\pi_\mathcal{A}} + q_{\pi^{-1}}) \cdot \ell). \tag{2}$$

Here, t_f is the time required to compute f, and space is measured in the number of storage bits.

Proof. See Appendix A □

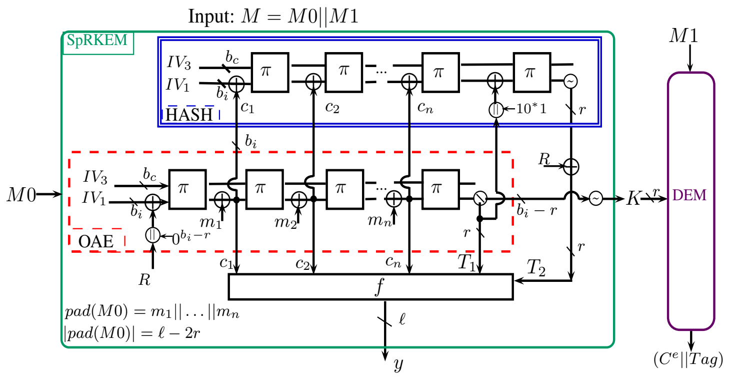
5 Sponge based KEM and Tag-KEM with partial message recovery

In this section, we present a *Sponge based key encapsulation mechanism with partial message recovery* (SpRKEM)(see Figure 5) and a *Sponge based Tag-Key encapsulation mechanism with partial message recovery* (Tag-SpRKEM) (Figure 6) using f-SpAEP.

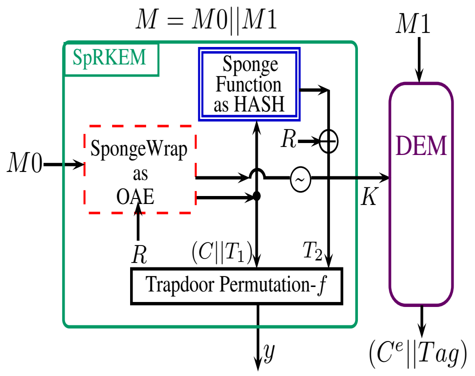
First, we briefly discuss about the KEMs (RKEM and Tag-RKEM) with partial message input/recovery introduced by Bjørstad et al. [8]. RKEM uses a *IND-CCA* secure public key encryption to transmit a symmetric key and some partial message data in secure manner. [8] shows, during encapsulation, RKEM takes a partial message $M0$. Using a public key encryption($\mathcal{E}$) and a hash function H, it computes key encapsulation $y = \mathcal{E}(pk, M0, R)$ for some randomness R and a symmetric key $K = H(M0||R)$ and returns (K, y). During decapsulation, RKEM takes y as a input and decrypts it using decryption algorithm($\mathcal{D}$) corresponding to $\mathcal{E}$. $\mathcal{D}(sk, y)$ outputs either $\perp$ and halts or the pair $(M0, R)$ and the algorithm proceeds to compute $K = H(M0||R)$ and returns $(M0, K)$. [8] suggested that any public key encryption schemes tat are securely randomness recovering, like OAEP-based schemes, are suitable for replacing $\mathcal{E}$ in practice. Usage of a extra H in RKEM enables the scheme to run RKEM and DEM independently and in parallel, due to independent $\mathcal{E}$ and H, during the encryption.

Now, we look at SpRKEM, Figure 5, we have similar structure like of the RKEM. In SpRKEM, f-SpAEP takes a partial message $M0$ and outputs y and also outputs K. During decapsulation, SpRKEM takes y as input and return either $\perp$ or $(M0, K)$ directly. In SpRKEM, SpongeWrap as *OAE* inherit the H used in the RKEM and produces a random symmetric key K. Therefore,f-SpAEP, already proven CCA secure, can directly replace the $\mathcal{E}$ required in the RKEM of [8] and does not require any other extra hash function (H) like in the RKEM of [8] and in between $\mathcal{D}$ and DEM during the decryption.

If we have one extra hash function H in SpRKEM, like required in RKEM, then SpRKEM and DEM can also be used independently in parallel, like in RKEM. But during decryption, extra call of H comes in between SpRKEM and DEM which is not the case earlier in SpRKEM, without H. Security requirement for DEM in RKEM/DEM is *IND-PA* and *INT-CTXT*, defined in [8], remains same for DEM in SpRKEM/DEM composition.



(a) SpRKEM/DEM detailed view



(b) SpRKEM/DEM general view

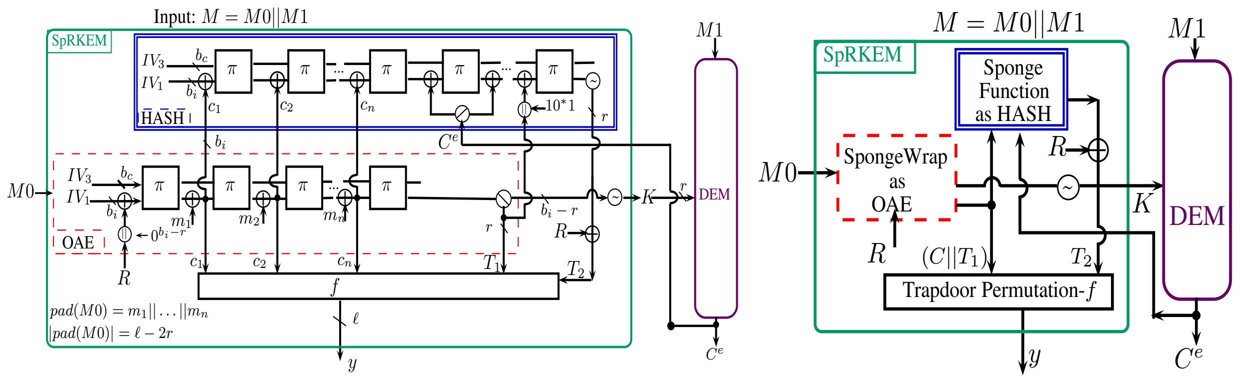
Figure 5: SpRKEM: f-SpAEP as RKEM version. (a) SpAEP with any trapdoor one way permutation f and public invertible permutation $\pi : \{0,1\}^{b_i} \times \{0,1\}^{b_c} \leftarrow \{0,1\}^{b_i} \times \{0,1\}^{b_c}$. From message $M = M0||M1$, f-SpAEP as SpRKEM accepts message $M0$ and output y and session key K. Internally SpAEP, OAE call pad($M0$)=$m_1||\dots||m_n$ such that $n = (\ell - 2r)/b_i$, $|pad(M0)| = (\ell - 2r)$ and each message bock is of length b_i where ℓ is size of trapdoor permutation-f and $|R| = |T_1| = |T_2| = r$. The symbol ⧀ represent taking r-bit from b_i-bit input. The symbol ⦶ represent concatenation. The symbol ⦸ represents a function that divide b_i-bit input into r-bit and $(b_i - r)$-bit value. Rest of the message $M1$ is processed by DEM using session key $K \in \{0,1\}^r$ and output $(C^e||Tag)$.

In Tag-KEM, KEM is used to preserve the integrity of the ciphertext, produced by DEM, in addition to confidentiality of symmetric key K. Tag-RKEM/DEM [8], uses an extra hash function KDF in addition to the $\mathcal{E}$ and H required in RKEM. During encryption, after computing symmetric key K, Tag-RKEM takes partial ciphertext, computed by DEM, as input tag and then computes key encapsulation. During decryption, Tag-RKEM recovers the partial ciphertext instead of partial message. Therefore, Tag-RKEM is consider as *Tag-KEM with partial ciphertext recovery*. Tag-RKEM/DEM requires a *IND-CCA* secure Tag-RKEM and *INP-PA* secure DEM.

f-OAEP-4X has simpled the Tag-RKEM structure, where f is a trapdoor one-way permutation. f-OAEP-4X take input of partial message $M0$ and computes the symmetric key K then for computing encapsulation f-OAEP-4X also take ciphertext τ as tag, generated by DEM using K, during encryption. During decryption, f-OAEP-4X ables to recover the partial message. Therefore, OAEP-4X with any trapdoor one-way permutation used as *Tag-KEM with partial message recovery*, where partial message can be recovered directly from Tag-KEM. Similar to Tag-RKEM/DEM, f-OAEP-4X is *IND-CCA* secure and require *INP-PA* secure symmetric encryption scheme as DEM.

The Tag-SpRKEM/DEM, Figure 6, follows the same idea of Tag-KEM with partial message recovery (OAEP-4X) but SpAEP of Tag-SpRKEM have its own advantage over OAEP-4X, discussed already. CCA secure f-SpAEP takes a partial message $M0$ and computes a symmetric key K, then takes ciphertext (C^e), generated by *IND-PA* secure DEM, to maintain integrity of the ciphertext. Tag-SpRKEM outputs the encapsulation of K and encryption of $M0$ as single output y. Tag-SpRKEM/DEM is very similar to CCA secure f-SpAEP processing long messages, only the computations done by SpongeWrap/*OAE* over extra message $M1$ is done by DEM part of

Tag-SpRKEM/DEM.



(a) Tag-SpRKEM/DEM detailed view (b) Tag-SpRKEM/DEM general view

Figure 6: Tag-SpRKEM: f-SpAEP as Tag-RKEM version. (a) SpAEP with any trapdoor one way permutation f and public invertible permutation $\pi : \{0,1\}^{b_i} \times \{0,1\}^{b_c} \leftarrow \{0,1\}^{b_i} \times \{0,1\}^{b_c}$. SpAEP accepts message $M0$ and internally OAE call pad($M0$)=$m_1|| \dots ||m_n$ such that $n = (\ell - 2r)/b_i$, $|pad(M0)| = (\ell - 2r)$ and each message bock is of length b_i where ℓ is size of trapdoor permutation-f and $|R| = |T_1| = |T_2| = r$. OAE output the session key K for DEM. The symbol ⦾ represent taking r-bit from b_i-bit input. The symbol ⦶ represent concatenation. The symbol ⦸ represent a function that divide b_i-bit input into r-bit and $b_i - r$-bit value. The symbol ⊘ represent a function dividing input into b_i-bit blocks. Rest of the message $M1$ is processed by DEM using session key K and output (C^e). This C^e is a input to Hash part of SpAEP and SpRKEM outputs y.

6 Conclusion

We presented a new variant, SpAEP, of OAEP using Sponge constructions that does not require hash output of arbitrary length, whereas all previous OAEP based encryption proven secure in random-oracle model require one or more hash output of arbitrary length. Versatility of Sponge construction helps us to reduce number of round function as compared to previous OAEP-type schemes (OAEP, OAEP+, OAEP-3R, OAEP-4X) and in constructing KEMs that require a PKC scheme with ability of randomness recovery. Ability of handling long messages enables the use of SpAEP with any trapdoor one-way permutation as hybrid encryption.

References

[1] Mihir Bellare and Phillip Rogaway. Optimal Asymmetric Encryption. In Alfredo De Santis, editor, *EUROCRYPT*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer, 1994.

[2] Duong Hieu Phan and David Pointcheval. Chosen-Ciphertext Security without Redundancy. In Chi-Sung Laih, editor, *Advances in Cryptology - ASIACRYPT 2003, 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 - December 4, 2003, Proceedings*, volume 2894 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2003.

[3] Duong Hieu Phan and David Pointcheval. OAEP 3-Round: A Generic and Secure Asymmetric

Encryption Padding. In Pil Joong Lee, editor, *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*, pages 63–77. Springer, 2004.

[4] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. Relaxing Chosen-Ciphertext Security. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 565–582. Springer, 2003.

[5] Masayuki Abe, Eike Kiltz, and Tatsuaki Okamoto. Chosen Ciphertext Security with Optimal Ciphertext Overhead. *IEICE Transactions*, 93-A(1):22–33, 2010.

[6] RSA Laboratories. PKCS #1 v2.1: RSA cryptography standard, June 2002.

[7] Masayuki Abe, Rosario Gennaro, and Kaoru Kurosawa. Tag-KEM/DEM: A New Framework for Hybrid Encryption. *J. Cryptology*, 21(1):97–130, 2008.

[8] Tor E. Bjørstad, Alexander W. Dent, and Nigel P. Smart. Efficient KEMs with Partial Message Recovery. In Steven D. Galbraith, editor, *Cryptography and Coding, 11th IMA International Conference, Cirencester, UK, December 18-20, 2007, Proceedings*, volume 4887 of *Lecture Notes in Computer Science*, pages 233–256. Springer, 2007.

[9] Ronald Cramer and Victor Shoup. Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. *IACR Cryptology ePrint Archive*, 2001:108, 2001. `http://eprint.iacr.org/2001/108`.

[10] Kaoru Kurosawa and Yvo Desmedt. A New Paradigm of Hybrid Encryption Scheme. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2004.

[11] Alexander W. Dent. A Designer's Guide to KEMs. In Kenneth G. Paterson, editor, *Cryptography and Coding, 9th IMA International Conference, Cirencester, UK, December 16-18, 2003, Proceedings*, volume 2898 of *Lecture Notes in Computer Science*, pages 133–151. Springer, 2003.

[12] Eike Kiltz. Chosen-Ciphertext Security from Tag-Based Encryption. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*, pages 581–600. Springer, 2006.

[13] Joonsang Baek, Willy Susilo, Joseph K. Liu, and Jianying Zhou. A New Variant of the Cramer-Shoup KEM Secure against Chosen Ciphertext Attack. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *Applied Cryptography and Network Security, 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2-5, 2009. Proceedings*, volume 5536 of *Lecture Notes in Computer Science*, pages 143–155, 2009.

[14] Tatsuaki Okamoto. Authenticated Key Exchange and Key Encapsulation in the Standard Model. In Kaoru Kurosawa, editor, *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*, volume 4833 of *Lecture Notes in Computer Science*, pages 474–484. Springer, 2007.

[15] Dennis Hofheinz and Eike Kiltz. Secure Hybrid Encryption from Weakened Key Encapsulation. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, volume 4622 of *Lecture Notes in Computer Science*, pages 553–571. Springer, 2007.

[16] Mihir Bellare and Phillip Rogaway. Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS '93, pages 62–73, New York, NY, USA, 1993. ACM.

[17] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures - how to sign with RSA and rabin. In Ueli M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer, 1996.

[18] Jean-Sébastien Coron. On the Exact Security of Full Domain Hash. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235. Springer, 2000.

[19] Gaëtan Leurent and Phong Q. Nguyen. How Risky Is the Random-Oracle Model? In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, volume 5677 of *Lecture Notes in Computer Science*, pages 445–464. Springer, 2009.

[20] Michael Peeters Guido Bertoni, Joan Daemen and Gilles Van Assche. Sponge functions, 2007. ECRYPT Hash Function Workshop.

[21] Michael Peeters Guido Bertoni, Joan Daemen and Gilles Van Assche. Permutation-based encryption, authentication and authenticated encryption. *Directions in Authenticated Ciphers*, 2012.

[22] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography*, volume 7118 of *Lecture Notes in Computer Science*, pages 320–337. Springer, 2011.

[23] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. The Sponge Functions Corner. `http://sponge.noekeon.org/`.

[24] Elena Andreeva, Begül Bilgin, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. APE: Authenticated Permutation-Based Encryption for Lightweight Cryptography. In Carlos Cid and Christian Rechberger, editors, *FSE*. Springer, 2014.

[25] Competition for Authenticated Encryption: Security, Applicability, and Robustness(CAESAR), 2014. `https://competitions.cr.yp.to/caesar.html`.

[26] Password Hashing Competition (PHC), 2014. `https://password-hashing.net/index.html`.

[27] Paul Baecher, Christina Brzuska, and Arno Mittelbach. Reset Indifferentiability and Its Consequences. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT*

2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I, volume 8269 of *Lecture Notes in Computer Science*, pages 154–173. Springer, 2013.

[28] Arno Mittelbach. Salvaging Indifferentiability in a Multi-stage Setting. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 603–621. Springer, 2014.

[29] Victor Shoup. OAEP Reconsidered. *J. Cryptology*, 15(4):223–249, 2002.

[30] Chris Peikert. Lattice Cryptography for the Internet. In Michele Mosca, editor, *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014, Waterloo, ON, Canada, October 1-3, 2014. Proceedings*, volume 8772 of *Lecture Notes in Computer Science*, pages 197–219. Springer, 2014.

[31] Rikke Bendlin, Sara Krehbiel, and Chris Peikert. How to Share a Lattice Trapdoor: Threshold Protocols for Signatures and (H)IBE. In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *Applied Cryptography and Network Security - 11th International Conference, ACNS 2013, Banff, AB, Canada, June 25-28, 2013. Proceedings*, volume 7954 of *Lecture Notes in Computer Science*, pages 218–236. Springer, 2013.

[32] Daniele Micciancio and Chris Peikert. Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer, 2012.

[33] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer, 1998.

[34] Charles Rackoff and Daniel R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In Joan Feigenbaum, editor, *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer, 1991.

[35] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-Malleable Cryptography (Extended Abstract). In Cris Koutsougeras and Jeffrey Scott Vitter, editors, *STOC*, pages 542–552. ACM, 1991.

[36] Mihir Bellare and Phillip Rogaway. Code-Based Game-Playing Proofs and the Security of Triple Encryption. *IACR Cryptology ePrint Archive*, 2004:331, 2004.

[37] Mihir Bellare and Phillip Rogaway. The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006.

A Proof of Theorem 1

Proof. We will use Game based playing technique [36, 37]. We start from the original CCA game as defined in Section 2. $EX_{f-SpAEP,\mathcal{A}}$ Or $Exp^{ind-cca2-d}_{f-SpAEP,\mathcal{A}}(\ell) = d \mid d \xleftarrow{\$} \{0,1\}$ denote the event that $\mathcal{A}$ outputs $d' = d$ where $d \xleftarrow{\$} \{0,1\}$. We want to show that $\mid Pr[EX_{f-SpAEP,\mathcal{A}}] \mid = \frac{1}{2} + negl(\ell)$. We slightly change f-SpAEP into a sequence G0, G1, …, G12 such that:

$Pr[EX_{f-SpAEP,\mathcal{A}}] = Pr[EX_{G0,\mathcal{A}}]$
$Pr[EX_{G(i-1),\mathcal{A}}] = Pr[EX_{Gi,\mathcal{A}}] + negl(\ell)\ \forall 1 \leq i \leq 11$
$Pr[EX_{Gi,\mathcal{A}}] = \frac{1}{2}$

Each game has the following functions:

- *Encryption* (Enc), *Decryption*(Dec): perform Encryption and Decryption,
- π, π^{-1}: public invertible permutation and its inverse,
- π_{Enc}, π_{Dec}: permutation π calls by encryption and decryption functions,
- $\pi_{\mathcal{A}}$, $\pi^{-1}_{\mathcal{A}}$: permutation π, π^{-1} calls by adversary $\mathcal{A}$.

Encryption, *Decryption*, $\pi_{\mathcal{A}}$ and $\pi^{-1}_{\mathcal{A}}$ are public oracles, which are also accessible to the adversary. In each game, the following sets are maintained: I_π by π and π^{-1}, I_{Enc} by π_{Enc}, I_{Dec} by π_{Dec} and $I_{\mathcal{A}}$ by $\pi_{\mathcal{A}}$ and $\pi^{-1}_{\mathcal{A}}$ to store input-output relations.

Another set $Y : \{g : g \in \{0,1\}^{b_c}\}$ is also maintained internally by π and π^{-1} for storing capacity bits. The set Y is initialized to $\{IV_2, IV_3\}$ because IV_2 is the capacity part of the input to first π of OAE part and IV_3 is the capacity part of the input to the first π of *Hash* part. The set Y is updated on every call to π. Precisely, two b_c-bit values are appended to Y on each π call. These two values are the capacity bits of the inputs and output of π.

Note that $q = q_\pi + q_{\pi^{-1}}$, $q_\pi = q_{\pi_{\mathcal{A}}} + q_{\pi_{Enc}}$ and q_D= number of decryption queries. Further, the encryption query has $(2ne + 1)$ calls to π_{Enc}.

In each of the games G0, G1, G2, G3, G4, G5 we make small incremental changes in the permutation to make it ideal permutation. In games G6, G7, we make changes in the *Decryption* oracle and make it independent of f. Finally, in games G8, G9, G10, G11, G12 we make changes in *Encryption* oracle along with some changes in $\pi_{\mathcal{A}}$ oracle to achieve that d of M_d is independent of all previous queries. We represent the *Hash* part of SpAEP as a function $H^\pi(j_1, j_2, j_3 ..., j_i, j_{i+1})$ whose output J is such that

$J || * = \pi_{Enc}(\pi_{Enc}(\ldots(\pi_{Enc}(\pi_{Enc}(j_2||0^{b_c} \oplus j1) \oplus j_3||0^{b_c}) \oplus j_4||0^{b_c}) \ldots \oplus j_{i-1}||0^{b_c}) \oplus j_i||0^{b_c+b_r-r}) \oplus j_{i+1}||0^{b_c+b_r-r}$, where π is b-bit permutation, $j_1 \in \{0,1\}^b$, $(j_2, j_3, \ldots j_{i-1}) \in \{0,1\}^{b_i}$, $(J, j_i, j_{i+1}) \in \{0,1\}^r$ and $* \in \{0,1\}^{b_c}$.

Game G0: This game perfectly simulates the f-SpAEP.

$$\Pr[EX_{f-SpAEP,\mathcal{A}}]=\Pr[EX_{G0,\mathcal{A}}].$$

Game G0 and Game G1: The Code for both these games is exactly same. In the permutation, both games choose their response randomly while excluding the previous responses, in order to satisfy the permutation property.

$$\Pr[EX_{G0,\mathcal{A}}]=\Pr[EX_{G1,\mathcal{A}}].$$

Game G1 and Game G2: Both the games are same till **bad** occurs. The event **bad** occurs when a collision over b-bit outputs of permutation π takes place. If q is the total number of queries to π and π^{-1}, then Pr[**bad**] is $\leq \frac{q(q-1)}{2^{b+1}}$.

$$|Pr[EX_{G2,\mathcal{A}}] - Pr[EX_{G1,\mathcal{A}}]| = \Pr[\mathbf{bad}] \leq \frac{(q-1)q}{2^{b+1}}.$$

Game G2 and Game G3: Both the games are the same where the output of π is not checked for collision over previous responses.

$$|Pr[EX_{G3,\mathcal{A}}] = Pr[EX_{G2,\mathcal{A}}]|.$$

Game G3 and Game G4: Both the games are the same till **bad** occurs. The event **bad** occurs if there is a collision over c-bit output of permutation π. Pr[**bad**] is $\leq \frac{q(q+1)}{2^{b_c}}$.

$$|Pr[EX_{G4,\mathcal{A}}] - Pr[EX_{G3,\mathcal{A}}]| = \Pr[\mathbf{bad}] \leq \frac{q(q+1)}{2^{b_c}}.$$

Game G4 and Game G5: Both the games are the same.

$$|Pr[EX_{G5,\mathcal{A}}] = Pr[EX_{G4,\mathcal{A}}]|.$$

Game G5 and Game G6: Both the games are same. In Game G6 only a dummy operation of $flag \leftarrow new$ is added in the *Decryption* oracle to denote a new query. The query is new in the sense that neither the query nor any part of the query during internal calls to π_{Dec} was queried earlier by the adversary. That is, query $\notin I_{\mathcal{A}}$.

$$|Pr[EX_{G6,\mathcal{A}}] = Pr[EX_{G5,\mathcal{A}}]|.$$

Game G6 and Game G7: Both the games act similarly till **bad** occurs. The event **bad** occurs in *Decryption* oracle when a new query results in $T_1 = T_1'$ (mentioned in Algorithm 2 line 15). The **bad** event occurs with probability $\frac{5q_D}{2^r} + \frac{q_{\pi_{\mathcal{A}}} + q_{\pi^{-1}}}{2^r}$.

$$|Pr[EX_{G7,\mathcal{A}}] - Pr[EX_{G6,\mathcal{A}}]| = \Pr[\mathbf{bad}] \leq \frac{5q_D}{2^r} + \frac{q_{\pi_{\mathcal{A}}} + q_{\pi^{-1}}}{2^r}.$$

Let $(v_1||v_2) = \pi_{Dec}(x||w)$, where x,$v_1 \in \{0,1\}^{b_i}$ and w,$v_2 \in \{0,1\}^{b_c}$. In decryption, a input is a *new query* to π_{Dec} when $((x||w),(v_1||v_2)) \notin I_{\mathcal{A}}$ and *old query* when $((x||w),(v_1||v_2)) \in I_{\mathcal{A}}$. If a *new query* $(x||w)$ is input to π_{Dec} during decryption, then π_{Dec} outputs $v_1||v_2$, where $v_2 \notin Y$. That is, v_2 is also new. Since v_2 is unseen so far, it ensures that the input to the next call of π is certainly new. Further, since v_2 is new, next input $x'||v_2$ satisfies the condition $(x'||v_2, *) \notin I_{\mathcal{A}}$, where $*$ stands for any b bit value. Therefore one *new query* makes all subsequent inputs to $\pi_{Dec}(\cdot)$ as new. Any *new query* to π_{Dec} implies that a ciphertext y queried to *Decryption* oracle has never been generated by the adversary. In Game G7, *Decryption* oracle return **Invalid** whenever adversary makes such a query.

To know if a *new query* has been made in SpAEP *Decryption* oracle, we consider three check-points, called A, B and C in Figure A. Next we explain the situation when a **bad** event can occur in Game G7.

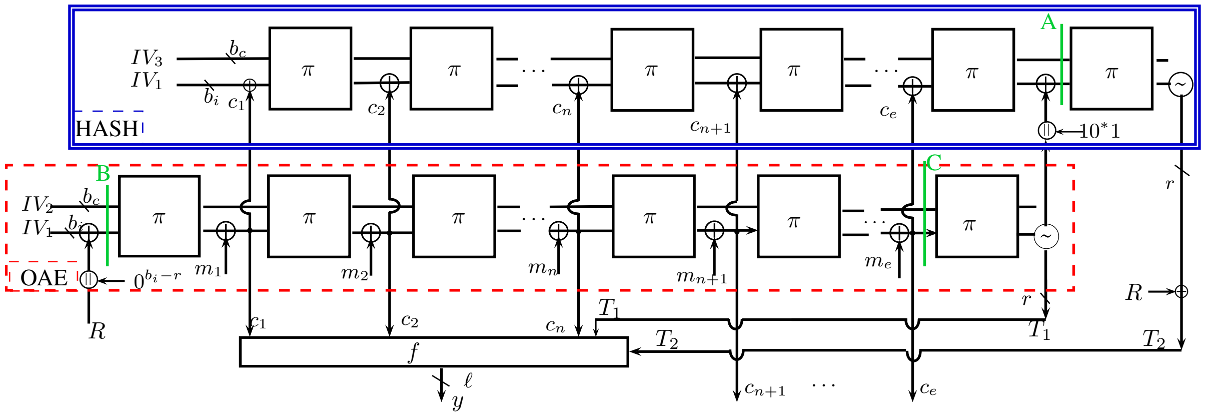


Figure 7: f-SpAEP with three checkpoints for testing if a new query has been made.

In *Hash*-part, if any input before A is new, then A is also new as explained earlier. Hence a decryption query is certainly new if A is new. In the case of checkpoints B and C, it is not possible that B is *new query* and C is *old query*. This follows from our discussion above. Therefore we only need to check C to determine if there is a *new query* in the OAE part.

During encryption, let us denote the values at checkpoints A, B and C by $\alpha, R^*||0^{b_r-r}||IV_2$ and β respectively. Let $y^*||C^{e*}$ be the target ciphertext and $C^* = C^{f*}||C^{e*}$ where $C^{f*} = c_1^*||\ldots||c_n^*||T_1^*||T_2^*$ and $C^{e*} = c_{n+1}^*||\ldots||c_e^*$ such that $y^* = f(C^{f*})$.
The following cases cover all the possible cases for *new query*.

CASE-1 (A *new*, B *new*, C *new*): The bad event occurs only when tag $T_1 = T_1'$ (shown in Algorithm 2 and Appendix A: game G7).

(a) C$\neq \beta$: Then $T_1 = T_1'$ implies collision of the outputs of π over r-bit value. Probability of this event is $\frac{q_D}{2^r}$ for q_D queries to *Decryption* oracle

(b) C$=\beta$: Then $T_1 = T_1^*$ which means $c_i = c_i^*$ for all i such that $1 \leq i \leq n$ and $R = R^*$. This in turn results in $T_2 = T_2^*$. This leads to $C = C^*$, which is not allowed because adversary can not query $y^* = f(C^*)$ to *Decryption* oracle.

CASE-2 (A *new*, B *new*, C *old*): This case is impossible. It is due to the fact that if B is new, then all subsequent inputs to π_{Dec} including C are also new.

CASE-3 (A *new*, B *old*, C *new*): The bad event occurs only when tag $T_1 = T_1'$ as in CASE-1. This happens with probability $\frac{q_D}{2^r}$.

CASE-4 (A *new*, B *old*, C *old*):

(a) A$\neq \alpha$: B and C are old queries in this case and hence R, T_1 is already known to the adversary. T_2 is also fixed due to the query $y||C^e$ to the *Decryption* oracle. Further, $\lfloor \pi(A) \rfloor_r = R \oplus T_2$ results in $T_1 = T_1'$, which is a collision of output of π(A) over r-bit value. Probability of this event is $\frac{q_D}{2^r}$ for q_D queries to the *Decryption* oracle.

(b) A$=\alpha$: This results in $T_2 = T_2^*$ due to the permutation property of π. This leads to $c_i = c_i^*$ for all i such that $1 \leq i \leq n$. If $R = R^*$, then $y = y^*$ which is not allowed. On the other hand, if $R \neq R^*$ and $T_1 = T_1'$, then the OAE part results in collision over r-bits. This is a kind of hash collision on outputs of OAE for different inputs. Probability of such a hash collision is $\frac{q_{\pi_{\mathcal{A}}}+q_\pi-1}{2^r}$.

CASE-5 (A *old*, B *new*, C *new*): The bad event occurs only when Tag $T_1 = T_1'$ as in CASE-1,3. This happen only with probability $\frac{q_D}{2^r}$.

CASE-6 (A *old*, B *new*, C *old*): This case is impossible. It is due the fact that if B is new, then all the subsequent inputs to π_{Dec} including C are also new.

CASE-7 (A *old*, B *old*, C *new*): Same as CASE-1,3,5.

CASE-8 (A *old*, B *old*, C *old*): The bad event can not occur in this case.

Game G7 and Game G8: Both the games are same. Game G7 and G8 both return **Invalid** when a *new query* is given to the *Decryption* oracle. In Game G8, a message M is returned only when all the input-output relations of π, which would be possible during the encryption of M, are already in $I_{\mathcal{A}}$. Game G8 iterates over all the possible pairs of (input,output) of $\pi \in I_{\mathcal{A}}$. This makes the *Decryption* oracle independent of f.

On query $y||C^e$, the *Decryption* oracle returns a valid M only if the adversary knows the plaintext-ciphertext pair $(M, y||C^e)$; otherwise it returns **Invalid**.

$$|Pr[EX_{G8,\mathcal{A}}] = Pr[EX_{G7,\mathcal{A}}]|.$$

Game G8 and Game G9: We start incremental changes in *Encryption* oracle from Game G9. In G8, R^* is generated randomly. In G9, R^* is computed using the value of randomly generated c_i^* $(1 \le i \le e)$ and T_1^*, T_2^* . The value of R^* is calculated via $H^{\pi_{Enc}}(IV, c_1^*, c_2^*, \ldots, c_e^*, T_1^*) \oplus T_2^*$. Since π is an ideal permutation and T_2^* is a random value, R^* will also be random. Therefore, G8 and G9 are same.

$$|Pr[EX_{G9,\mathcal{A}}] = Pr[EX_{G8,\mathcal{A}}]|.$$

Game G9 and Game G10: In the game G10, R^* is generated in same way as in G9. In *Encryption* oracle, π is a ideal permutation which results in random $c_i^*(1 \le i \le n)$. Therefore, in G10, the values of c_i for all i are replaced by random values c_i^*. Similarly T_1 is replaced with T_1^*. Due to initial random R^*, $T_2 = T_2^*$. Both games G9 and G10 will behave the same way until 'Bad'. The Bad event occurs when the adversary queries $R^*||0^{b_r-r}||IV_2$ to $\pi_{\mathcal{A}}$ or receives response $R^*||0^{b_r-r}||IV_2$ from $\pi_{\mathcal{A}^{-1}}$. In G10, R^* is calculated using C^*, unlike C using the R^* as in G9. In G9, relation between $c_1, c_2, \ldots, c_e$ is generated by R^*. However, relation between $c_1^*, c_2^*, \ldots, c_e^*$ does not exist in G10. This gap in the relation can be exploited by the adversary if adversary queries $R^*||0^{b_r-r}||IV_2$ to $\pi_{\mathcal{A}}$ or receives response $R^*||0^{b_r-r}||IV_2$ from $\pi_{\mathcal{A}^{-1}}$.

$$|Pr[EX_{G10,\mathcal{A}}] - Pr[EX_{G9,\mathcal{A}}]| = \text{Pr[Bad]}.$$

Game G10 and Game G11: Both the games are same. In game G10, $C^* = C$. In G11, *Encryption* oracle replaces C^* with randomly generated C, without calling π_{Enc}. Both games G10 and G11 return a random $y^*||C^e = f(C^{f*} = C^f)||C^e$. The Bad event in G10 takes place in G11 as Bad1.

$$|Pr[EX_{G11,\mathcal{A}}] = Pr[EX_{G10,\mathcal{A}}]|, \text{ Pr[Bad]=Pr[Bad1]}.$$

Game G11 and Game G12: Game G12 is the final game of adversary $\mathcal{A}$. From G11, a random y is the output of *Encryption* oracle and C is unknown to adversary independent of M_b. Therefore, if a random C is given to the $\mathcal{A}$ in G12, then R^* will be unknown to the adversary. Bad1 event in G10 is same as Bad2 in G11.

$$|Pr[EX_{G12,\mathcal{A}}] = Pr[EX_{G11,\mathcal{A}}]|, \text{ Pr[Bad1]=Pr[Bad2]}.$$

If a random C is given to the $\mathcal{A}$ in G12, then R^* will be unknown to the adversary and C will be independent of d of M_d Therefore,

$$|Pr[EX_{G12,\mathcal{A}}] = \tfrac{1}{2}.$$

Given a target ciphertext y, Adversary $\mathcal{B}_{\mathcal{A}}$ uses $\mathcal{A}$ as a black box, while $\mathcal{A}$ uses G12.

A detailed description of the games and adversary $\mathcal{B}$ is given in Appendix B. The probability of Bad2 is as follows.

$$\begin{aligned}\text{Pr[Bad2]} &= \text{Pr}[R^*||0^{b_r-r}||IV_2 \text{ is queried to } (\pi_{\mathcal{A}} \text{ or } \pi_{\mathcal{A}}^{-1})] \\ &= \text{Pr}[(R^*||0^{b_r-r}||IV_2 \text{ is queried to } (\pi_{\mathcal{A}} \text{ or } \pi_{\mathcal{A}}^{-1})) \wedge (I_{Enc} \subset I_{\mathcal{A}})] \\ &\quad + \text{Pr}[(R^*||0^{b_r-r}||IV_2 \text{ is queried to } (\pi_{\mathcal{A}} \text{ or } \pi_{\mathcal{A}}^{-1})) \wedge (I_{Enc} \not\subset I_{\mathcal{A}})].\end{aligned}$$

$(I_{Enc} \subset I_{\mathcal{A}})$ implies that all the input-output relations of π_{Enc} are also known to the adversary $\mathcal{A}$ via set $I_{\mathcal{A}}$. Therefore $\mathcal{A}$ knows all c_i^* for $1 \leq i \leq e$ and T_1^*. Moreover, the adversary $\mathcal{A}$ learns T_2^* if it is allowed to query $R^*||0^{b_r-r}||IV_2$.

Given $y||C^e$, if $R^*||0^{b_r-r}||IV_2$ is queried to $(\pi_{\mathcal{A}}$ or $\pi_{\mathcal{A}}^{-1})$, then it reveals C completely. Therefore,

$$\text{Pr[Bad2]} = Adv_f^{owtp}(\mathcal{B}_{\mathcal{A}}) + \text{Pr}[(R^*||IV_2 \text{ is queried to } (\pi_{\mathcal{A}} \text{ or } \pi_{\mathcal{A}}^{-1})) \wedge (I_{Enc} \not\subset I_{\mathcal{A}})].$$

$I_{Enc} \not\subset I_{\mathcal{A}}$ implies that one of the inputs to $H^{\pi_{Enc}}()$ is unknown to the adversary $\mathcal{A}$. It results in unknown output value from $H^{\pi_{Enc}}()$. Since T_2 is already random therefore R^* remains unknown and random to $\mathcal{A}$. Therefore, $R^*||IV_2$ query to $\pi_{\mathcal{A}}$ is equivalent to random guessing of R^*.

$$\text{Pr[Bad2]}=Adv_f^{owtp}(\mathcal{B}_{\mathcal{A}})+\tfrac{(q_{\pi_{\mathcal{A}}}+q_{\pi^{-1}})}{min(2^r,2^c)}.$$

From Definition 1 if f is trapdoor, then $Adv_f^{owtp}(\mathcal{B}_{\mathcal{A}}) \leq negl.$

The time and space complexities mentioned in Equation 1 and 2 are easy to verify.

This completes the proof of Theorem 1. □

B Games

Game G0: Initialise $I_\pi = I_{Enc} = I_{Dec} = I_{\mathcal{A}} = \emptyset$, $f : \{0,1\}^\ell \to \{0,1\}^\ell$, $IV_1 = 0^{b_i}$ $IV_2 = 0^{b_c}, IV_3 = IV_2 \oplus 1$

```
On Encryption-Query $M_d (d \xleftarrow{\$} \{0,1\})$
 1. Random Nonce: $R \xleftarrow{\$} \{0,1\}^r$
 2. $pad(M) = m_1||m_2||\ldots||m_e$, where
    $|m_i| = b_i \ \forall 1 \leq i \leq e$
 3. $x = x \oplus R||0^{b_r - r}$
 4. for $i = 1 \to e$ do
        $(x||w) = \pi(x||w)$
        $x = x \oplus m_i$
        $c_i = x$
 5. $(x||w) = \pi(x||w)$; $T_1 = \lfloor x \rfloor_r$
 6. $x = IV_1$ and $w = IV_3$
 7. for $i = 1 \to e$ do
        $x = x \oplus c_i$
        $(x||w) = \pi(x||w)$
 8. $x = x \oplus T_1||0^{b_r - r}$
 9. $(x||w) = \pi(x||w)$
10. $T_2 = \lfloor x \rfloor_r \oplus R$
11. $C^f = c_1||c_2||\ldots||c_n||T_1||T_2$
12. $C^e = c_{n+1}||\ldots||c_e$
13. $y = f(C^f)$
14. Return: $y||C^e$
```

```
On Decryption-Query $y||C^e$
 1. $C^f = c_1||c_2||\ldots||c_n||T_1||T_2 = f^{-1}(y)$;
    $c_{n+1}||\ldots||c_e = C^e$
 2. $C = c_1||c_2||\ldots||c_n||T_1||T_2||c_{n+1}||\ldots||c_e$,
    where $|c_i| = b_i, |T_1| = |T_2| = r$ for $1 \leq i \leq e$
 3. for $i = 1 \to e$ do
        $x = x \oplus c_i$
        $(x||w) = \pi(x||w)$
 4. $x = x \oplus T_1||0^{b_r - r}$
 5. $(x||w) = \pi(x||w)$; $R = \lfloor x \rfloor_r \oplus T_2$
 6. $x = R||0^{b_r - r}$; $w = IV_2$
 7. for $i = 1 \to e$ do
        $(x||w) = \pi(x||w)$
        $m_i = x \oplus c_i$
        $x = c_i$
 8. $(x||w) = \pi(x||w)$; $T_1' = \lfloor x \rfloor_r$
 9. if $T_1 = T_1'$ then
        if $\exists\ M$ s.t. $M = unpad(m_1||\ldots||m_e)$
        then
            Return:$M$
        else
            Return: Invalid
    else
        Invalid.
```

```
On $\pi$-Query $m$, where $m \in \{0,1\}^b$
 1. let $(x||w) = m$, where $x \in \{0,1\}^r$,
    $w \in \{0,1\}^c$,
 2. if $(m, v) \in I_\pi$ then return $v$
 3. $v \xleftarrow{\$} \{0,1\}^b$
 4. if $\exists\ m'$ s.t $(m', v) \in I_\pi$, then
    $v \xleftarrow{\$} \{0,1\}^b \setminus \{v : (*, v) \in I_\pi\}$, where
    $* \in \{0,1\}^b$
 5. $I_\pi = I_\pi \bigcup \{(m, v)\}$
 6. return $v$;
```

```
On $\pi^{-1}$-Query $v = \{v_1||v_2\}$. where
$v_1 \in \{0,1\}^r,\ v_2 \in \{0,1\}^c,\ v \in \{0,1\}^b$
 1. if $(m, v) \in I_\pi$ then return $m$
 2. $m \xleftarrow{\$} \{0,1\}^b$
 3. if $\exists\ v'$ s.t $(m, v') \in I_\pi$, then
    $m \xleftarrow{\$} \{0,1\}^b \setminus \{m : (m, *) \in I_\pi\}$, where
    $* \in \{0,1\}^b$
 4. $I_\pi = I_\pi \bigcup \{(m, v)\}$
 5. return $m$;
```

```
On $\pi_{Enc}$-Query $m$, where $m \in \{0,1\}^b$
 1. $v = \pi(m)$
 2. $I_{Enc} = I_{Enc} \bigcup \{(m, v)\}$
 3. return $v$;
```

```
On $\pi_{Dec}$-Query $m$, where $m \in \{0,1\}^b$
 1. $v = \pi(m)$
 2. $I_{Dec} = I_{Dec} \bigcup \{(m, v)\}$
 3. return $v$;
```

```
On $\pi_{\mathcal{A}}$-Query $v$, where $m \in \{0,1\}^b$
 1. $v = \pi(m)$
 2. $I_{\mathcal{A}} = I_{\mathcal{A}} \bigcup \{(m, v)\}$
 3. return $v$;
```

```
On $\pi_{\mathcal{A}}^{-1}$-Query $v$, where $v \in \{0,1\}^b$
 1. $m = \pi^{-1}(v)$
 2. $I_{\mathcal{A}} = I_{\mathcal{A}} \bigcup \{(m, v)\}$
 3. return $v$;
```

Figure 8: Game G0

Game $\boxed{\textbf{G1}}$ and Game G2 : Initialise $I_\pi = I_{Enc} = I_{Dec} = I_{\mathcal{A}} = \emptyset,\ f : \{0,1\}^\ell \rightarrow \{0,1\}^\ell,\ IV_1 = 0^{b_i}\ IV_2 = 0^{b_c}, IV_3 = IV_2 \oplus 1$

On Encryption-Query $M_d(d \xleftarrow{\$} \{0,1\})$
Same as Game 0

On Decryption-Query $y||C^e$
Same as Game 0

On π-Query m,where $m \in \{0,1\}^b$

```
1. let $(x||w)$=$m$,where $x \in \{0,1\}^r$, $w \in \{0,1\}^c$,
2. if $(m,v) \in I_\pi$ then return $v$
3. $v \xleftarrow{\$} \{0,1\}^b$
4. if $\exists\ m'$ s.t $(m',v) \in I_\pi$, then bad$\leftarrow$true and
   $\boxed{v \xleftarrow{\$} \{0,1\}^b \setminus \{v : (*,v) \in I_\pi\}}$, where
   $* \in \{0,1\}^b$
5. $I_\pi = I_\pi \bigcup \{(m,v)\}$
6. return $v$;
```

On π^{-1}-Query v, where $v \in \{0,1\}^b$

```
1. let $(v_1||v_2)$=$m$,where $v_1 \in \{0,1\}^r$,
   $v_2 \in \{0,1\}^c$,
2. if $(m,v) \in I_\pi$ then return $m$
3. $m \xleftarrow{\$} \{0,1\}^b$
4. if $\exists\ v'$ s.t $(m,v') \in I_\pi$, then bad$\leftarrow$true and
   $\boxed{m \xleftarrow{\$} \{0,1\}^b \setminus \{m : (m,*) \in I_\pi\}}$, where
   $* \in \{0,1\}^b$
5. $I_\pi = I_\pi \bigcup \{(m,v)\}$
6. return $m$;
```

On π_{Enc}-Query m, where $m \in \{0,1\}^b$
Same as Game 0

On π_{Dec}-Query m, where $m \in \{0,1\}^b$
Same as Game 0

On $\pi_{\mathcal{A}}$-Query m, where $m \in \{0,1\}^b$
Same as Game G0

On $\pi_{\mathcal{A}}^{-1}$-Query v, where $v \in \{0,1\}^b$
Same as Game0

Figure 9: Game G1 and Game G2

Game G3 and Game $\boxed{\textbf{G4}}$: Initialise $I_\pi = I_{Enc} = I_{Dec} = I_{\mathcal{A}} = \emptyset,\ f : \{0,1\}^\ell \rightarrow \{0,1\}^\ell,\ IV_1 = 0^{b_i}\ IV_2 = 0^{b_c},\ IV_3 = IV_2 \oplus 1,\ Y = \{IV_2, IV_3\}$

On Encryption-Query $M_d(d \xleftarrow{\$} \{0,1\})$
Same as Game G0

On Decryption-Query $y||C^e$
Same as Game G0

On π-Query m, where $m \in \{0,1\}^b$

```
1. let $(x||w)$=$m$,where $x \in \{0,1\}^r$, $w \in \{0,1\}^c$,
2. if $(m,v) \in I_\pi$ then return $v$
3. $v_1||v_2 \xleftarrow{\$} \{0,1\}^b$,where $v_1 \in \{0,1\}^r$,
   $v_2 \in \{0,1\}^c$,
4. if $v_2 \in Y \bigcup \{w\}$, then bad$\leftarrow$true and
   $\boxed{v_2 \xleftarrow{\$} \{0,1\}^c \setminus Y \bigcup \{w\}}$
5. $I_\pi = I_\pi \bigcup \{(m, v_1||v_2)\}$ and $Y = Y \bigcup \{v_2, w\}$
6. return $v = v_1||v_2$;
```

On π^{-1}-Query v. where $v \in \{0,1\}^b$

```
1. let $(v_1||v_2)$=$v$,where
   $v_1 \in \{0,1\}^r$, $v_2 \in \{0,1\}^c$,
2. if $(m,v) \in I_\pi$ then return $m$
3. $m'||m'' \xleftarrow{\$} \{0,1\}^b$,where $m' \in \{0,1\}^r$,
   $m'' \in \{0,1\}^c$,
4. if $m'' \in Y \bigcup \{v_2\}$, then bad$\leftarrow$true and
   $\boxed{m'' \xleftarrow{\$} \{0,1\}^c \setminus Y \bigcup \{v_2\}}$
5. $I_\pi = I_\pi \bigcup \{(m'||m'', v)\}$ and
   $Y = Y \bigcup \{m'', v_2\}$
6. return $m = m'||m''$;
```

On π_{Enc}-Query m, where $m \in \{0,1\}^b$
Same as Game 0

On π_{Dec}-Query m, where $m \in \{0,1\}^b$
Same as Game 0

On $\pi_{\mathcal{A}}$-Query m, where $m \in \{0,1\}^b$
Same as Game G0

On $\pi_{\mathcal{A}}^{-1}$-Query v, where $v \in \{0,1\}^b$
Same as Game0

Figure 10: Game G3 and Game G4

Game G5 Initialise $I_\pi = I_{Enc} = I_{Dec} = I_{\mathcal{A}} = \emptyset$, $f : \{0,1\}^\ell \rightarrow \{0,1\}^\ell$, $IV_1 = 0^{b_i}, IV_2 = 0^{b_c}, IV_3 = IV_2 \oplus 1,\ Y = \{IV_2, IV_3\}$

On Encryption-Query Same as Game G0

On Decryption-Query Same as G0

```
On π-Query m, where $m \in \{0,1\}^b$
1. let $(x||w)=m$, where $x \in \{0,1\}^r$, $w \in \{0,1\}^c$,
2. if $(m,v) \in I_\pi$ then return $v$
3. $v_1||v_2 \xleftarrow{\$} \{0,1\}^b$, where $v_1 \in \{0,1\}^r$, $v_2 \in \{0,1\}^c$,
4. if $v_2 \in Y \bigcup \{w\}$, then $v_2 \xleftarrow{\$} \{0,1\}^c \setminus Y \bigcup \{w\}$
5. $I_\pi = I_\pi \bigcup \{(m, v_1||v_2)\}$ and $Y = Y \bigcup \{v_2, w\}$
6. return $v = v_1||v_2$;
```

```
On π^{-1}-Query v. where $v \in \{0,1\}^b$
1. let $(v_1||v_2)=v$, where $v_1 \in \{0,1\}^r$, $v_2 \in \{0,1\}^c$,
2. if $(m,v) \in I_\pi$ then return $m$
3. $m'||m'' \xleftarrow{\$} \{0,1\}^b$, where $m' \in \{0,1\}^r$,
   $m'' \in \{0,1\}^c$
4. if $m'' \in Y \bigcup \{v_2\}$, then $m'' \xleftarrow{\$} \{0,1\}^c \setminus Y \bigcup \{v_2\}$
5. $I_\pi = I_\pi \bigcup \{(m'||m'', v)\}$ and $Y = Y \bigcup \{m'', v_2\}$
6. return $m = m'||m''$;
```

On π_{Enc}-Query m, where $m \in \{0,1\}^b$
Same as Game 0

On π_{Dec}-Query m, where $m \in \{0,1\}^b$
Same as Game 0

On $\pi_{\mathcal{A}}$-Query m, where $m \in \{0,1\}^b$ Same as G0

On $\pi_{\mathcal{A}}^{-1}$-Query v, where $v \in \{0,1\}^b$ Same as Game0

Figure 11: Game G5

$\boxed{\textbf{Game G6}}$ and $\boxed{\textbf{Game G7}}$: Initialise $I_\pi = I_{Enc} = I_{Dec} = I_{\mathcal{A}} = \emptyset$, $f : \{0,1\}^\ell \rightarrow \{0,1\}^\ell$, $IV_1 = 0^{b_i}$ $IV_2 = 0^{b_c}, IV_3 = IV_2 \oplus 1,\ Y = \{IV_2, IV_3\}$

On Encryption-Query Same as G0
On π-Query Same as G5
On π^{-1}-Query Same as G5
On π_{Enc}-Query Same as Game 0
On $\pi_{\mathcal{A}}$-Query Same as Game G0
On $\pi_{\mathcal{A}}^{-1}$-Query Same as Game0
On π_{Dec}-Query Same as Game 0

```
On Decryption-Query $y||C^e$
1. $C^f = c_1||c_2||\ldots||c_n||T_1||T_2 = f^{-1}(y)$; $c_{n+1}||\ldots||c_e = C^e$
2. $C = c_1||c_2||\ldots||c_n||T_1||T_2||c_{n+1}||\ldots||c_e$, where
   $|c_i| = b_i$, $|T_1| = |T_2| = r$ for $1 \le i \le n$
3. $x = IV_1$; $w = IV_3$; $flag \leftarrow old$
4. for $i = 1 \rightarrow e$ do
     $x = x \oplus c_i$
     if $\nexists\ v\ s.t.\ (x||w, v) \in I_{\mathcal{A}}$ then
       set $flag \leftarrow new$
     $(x||w) = \pi_{Dec}(x||w)$
5. if $\nexists\ v\ s.t.\ (x \oplus T_1||w, v) \in I_{\mathcal{A}}$ then
     set $flag \leftarrow new$
6. $(x||w) = \pi_{Dec}((x \oplus T_1||0^{b_r - r})||w)$
7. $R = \lfloor x \rfloor_r = v_1 \oplus T_2||0^{b_r - r}$; $w = IV_2$
8. for $i = 1 \rightarrow n$ do
     if $\nexists\ v\ s.t.\ (x||w, v) \in I_{\mathcal{A}}$ then
       set $flag \leftarrow new$
     $(x||w) = \pi_{Dec}(x||w)$
     $m_i = x \oplus c_i$
     $x = c_i$
9. if $\nexists\ v\ s.t.\ (x||w, v) \in I_{\mathcal{A}}$ then
     set $flag \leftarrow new$
10. $(x||w) = \pi_{Dec}(x||w)$, $T_1' = \lfloor x \rfloor_r$
11. if $T_1 = T_1'$ and $flag \leftarrow new$ then
      bad←true
      $\boxed{\text{Return:}M}$ $\boxed{\text{Return: }\textbf{Invalid}}$
12. if $T_1 = T_1'$ and $flag \leftarrow old$ then
      if $\exists\ M\ s.t.\ M = unpad(m_1||\ldots||m_e)$ then
        Return:$M$
      else
        Return: Invalid
    else
      Invalid.
```

Figure 12: Game G6 and Game G7

Game G8: Initialise $I_\pi = I_{Enc} = I_{Dec} = I_{\mathcal{A}} = \emptyset$, $f : \{0,1\}^\ell \to \{0,1\}^\ell$, $IV_1 = 0^{b_i}$ $IV_2 = 0^{b_c}, IV_3 = IV_2 \oplus 1$, $Y = \{IV_2\}$, $R^* \xleftarrow{\$} \{0,1\}^r$

Game G9: Initialise $I_\pi = I_{Enc} = I_{Dec} = I_{\mathcal{A}} = \emptyset$, $f : \{0,1\}^\ell \to \{0,1\}^\ell$, $IV_1 = 0^{b_i}$ $IV_2 = 0^{b_c}, IV_3 = IV_2 \oplus 1$, $Y = \{IV_2\}$, $R^* \xleftarrow{\$} \{0,1\}^r$
$C^* \xleftarrow{\$} \{0,1\}^{e*b_i+2r}$, Let $C^* = c_1^*||c_2^*||\dots||c_n^*||T_1^*||T_2^*||c_{n+1}^*||\dots||c_e^*$, where $|c^*_{[1,\dots,e]}| = b_i, |T^*_{[1,2]}| = r$
$R^*||* = \pi_{Enc}(\pi_{Enc}(\dots(\pi_{Enc}(\pi_{Enc}(c_1^*||0^{b_c} \oplus IV) \oplus c_2^*||0^{b_c}) \oplus c_3^*||0^{b_c})\dots \oplus c_e^*||0^{b_c}) \oplus T_1^*||0^{b_c+b_r-r}) \oplus T_2^*||0^{b_c+b_r-r}$

On Encryption-Query $M_d(d \xleftarrow{\$} \{0,1\})$

```
1. $pad(M) = m_1||m_2||\dots||m_e$, where
   $|m_i| = b_i\ \forall 1 \le i \le e$
2. $x = x \oplus R||0^{b_r-r}$
3. for $i = 1 \to e$ do
       $(x||w) = \pi(x||w)$
       $x = x \oplus m_i$
       $c_i^* = x$
4. $(x||w) = \pi(x||w)$; $T_1^* = \lfloor x \rfloor_r$
5. $x = IV_1$ and $w = IV_3$
6. for $i = 1 \to e$ do
       $x = x \oplus c_i^*$
       $(x||w) = \pi(x||w)$
7. $x = x \oplus T_1^*||0^{b_r-r}$
8. $(x||w) = \pi(x||w)$
9. $T_2^* = \lfloor x \rfloor_r \oplus R$
10. $C^{f*} = c_1^*||c_2^*||\dots||c_n^*||T_1^*||T_2^*$
11. $C^{e*} = c_{n+1}^*||\dots||c_e^*$
12. $y^* = f(C^{f*})$
13. Return: $y^*||C^{e*}$
```

On Decryption-Query $y||C^e$

```
1. $w_0 = IV_2$;
2. If $\exists$ pad$(M)=m_1||m_2||\dots||m_e$, $R$, after
   $x_0 = R||0^{b_r-r} \oplus IV_1$ such that
   $(x_0||w_0, v_{1_1}||v_{2_1}) \in I_{\mathcal{A}}$,
   $(x_n||w_n, v_{1_{e+1}}||v_{2_{e+1}}) \in I_{\mathcal{A}}$,
   $((z_{1_{e+1}} \oplus G)||u_{2_{e+1}}, z_{1_{e+2}}||z_{2_{e+2}}) \in I_{\mathcal{A}}$,
   $f(c_1||\dots||c_e||\lfloor v_{1_{e+1}} \rfloor_r||T_2) = y$, where
   $G = \lfloor v_{1_{e+1}} \rfloor_r||0^{b_r-r}$, $T_2 = \lfloor z_{1_{e+2}} \rfloor_r \oplus R$
   then return $M$
   else Return Invalid
```

On π-Query Same as Game G5
On π^{-1}-Query Same as Game G5
On π_{Enc}-Query m, where $m \in \{0,1\}^b$

On $\pi_{\mathcal{A}}$-Query m, where $m \in \{0,1\}^b$

```
1. $\boxed{\text{if } m = R^*||0^{b_r-r}||IV_2\ bad \leftarrow true}$
2. $v = \pi(m)$
3. $I_{\mathcal{A}} = I_{\mathcal{A}} \bigcup \{(m,v)\}$
4. return $v$;
```

On $\pi_{\mathcal{A}}^{-1}$-Query v, where $v \in \{0,1\}^b$

```
1. $m = \pi^{-1}(v)$
2. $\boxed{\text{if } m = R^*||0^{b_r-r}||IV_2\ bad \leftarrow true}$
3. $I_{\mathcal{A}} = I_{\mathcal{A}} \bigcup \{(m,v)\}$
4. return $v$;
```

Figure 13: Game G8

Following special notations is begin used during Game G8, G9, G10, G11, G12 and $\mathcal{B}_{\mathcal{A}}$ in decryption oracle:

1. During *OAE* part of SpAEP, we represent input-output relation of π's subsequent calls for $pad(M) = m_1||\dots||m_e$ by $(v_{1_{i+1}}||v_{2_{i+1}}) = \pi(x_i||w_i)$, where $x_i = v_{1_i} \oplus \{m_i\}$, $w_i = v_{2_i}$ $0 \le i \le e$, $v_{1_0} = IV_1$,$m_0 = R$, $w_0 = IV_2$, $v_{1_i}, x_i \in \{0,1\}^r$ and $v_{2_i}, w_i \in \{0,1\}^c$. Then c_i will represent $m_i \oplus v_{1_i}$, here $1 \le i \le e$.

2. Input-output relation of π's subsequent call during *Hash* part of SpAEP will be represented as follows: $(z_{1_i+1}||z_{2_i+1}) = \pi(u_{1_i}||u_{2_i})$,$u_{1_i} = c_i \oplus z_{1_i}$, $u_{2_i} = z_{2_i}$,where $1 \le i \le (e+2)$, $u_{2_1} = IV_3$, $z_{1_1} = IV_1$, $c_{e+1} = T_1$, $c_{e+2} = R$

Game G10: Initialise $I_\pi = I_{Enc} = I_{Dec} = I_\mathcal{A} = \emptyset$, $f : \{0,1\}^\ell \rightarrow \{0,1\}^\ell$, $IV_1 = 0^{b_i}$ $IV_2 = 0^{b_c}, IV_3 = IV_2 \oplus 1$, $Y = \{IV_2, IV_3\}$, $C^* \xleftarrow{\$} \{0,1\}^{e*b_i+2r}$, Let $C^* = c_1^*||c_2^*||\dots||c_n^*||T_1^*||T_2^*||c_{n+1}^*||\dots||c_e^*$, where $|c^*_{[1,\dots,e]}| = b_i, |T^*_{[1,2]}| = r$
$R^*||* = \pi_{Enc}(\pi_{Enc}(\dots(\pi_{Enc}(\pi_{Enc}(c_1^*||0^{b_c} \oplus IV) \oplus c_2^*||0^{b_c}) \oplus c_3^*||0^{b_c})\dots \oplus c_e^*||0^{b_c}) \oplus T_1^*||0^{b_c+b_r-r}) \oplus T_2^*||0^{b_c+b_r-r}$

On Encryption-Query $M_d(d \xleftarrow{\$} \{0,1\})$

```
1. $pad(M) = m_1||m_2||\dots||m_e$, where
   $|m_i| = b_i \; \forall 1 \leq i \leq e$
2. $x = x \oplus R^*||0^{b_r-r}$
3. for $i = 1 \rightarrow e$ do
       $(x||w) = \pi(x||w)$
       $x = x \oplus m_i$
       $c_i = x$
4. $(x||w) = \pi(x||w)$; $T_1 = \lfloor x \rfloor_r$
5. $x = IV_1$ and $w = IV_3$
6. for $i = 1 \rightarrow e$ do
       $x = x \oplus c_i$
       $(x||w) = \pi(x||w)$
7. $x = x \oplus T_1||0^{b_r-r}$
8. $(x||w) = \pi(x||w)$
9. $T_2 = \lfloor x \rfloor_r \oplus R$
10. $c_1 = c_1^*, \dots, c_e = c_e^*, T_1 = T_1^*, T_2 = T_2^*$
11. $C^f = c_1||c_2||\dots||c_n||T_1||T_2$
12. $C^e = c_{n+1}||\dots||c_e$
13. $y = f(C^f)$
14. Return: $y||C^e$
```

On Decryption-Query $y||C^e$
Same as Game G8
On π-Query m Same as Game G5
On π^{-1}-Query v Same as Game G5
On π_{Enc}-Query m Same as G0

On $\pi_\mathcal{A}$-Query m, where $m \in \{0,1\}^b$

```
1. if $m = R^*||0^{b_r-r}||IV_2$ $bad \leftarrow true$
2. $v = \pi(m)$
3. $I_\mathcal{A} = I_\mathcal{A} \bigcup \{(m,v)\}$
4. return $v$;
```

On $\pi_\mathcal{A}^{-1}$-Query v, where $v \in \{0,1\}^b$

```
1. $m = \pi^{-1}(v)$
2. if $m = R^*||0^{b_r-r}||IV_2$ $bad \leftarrow true$
3. $I_\mathcal{A} = I_\mathcal{A} \bigcup \{(m,v)\}$
4. return $v$;
```

Figure 14: Game G10

Game G11: Initialise $I_\pi = I_{Enc} = I_{Dec} = I_\mathcal{A} = \emptyset$, $f : \{0,1\}^\ell \rightarrow \{0,1\}^\ell$, $IV_1 = 0^{b_i}$ $IV_2 = 0^{b_c}, IV_3 = IV_2 \oplus 1$, $Y = \{IV_2, IV_3\}$, $C^* \xleftarrow{\$} \{0,1\}^{e*b_i+2r}$, Let $C^* = c_1^*||\dots||c_n^*||T_1^*||T_2^*||c_{n+1}^*||\dots||c_e^*$, where $|c^*_{[1,\dots,e]}| = b_i, |T^*_{[1,2]}| = r$
$R^*||* = \pi_{Enc}(\pi_{Enc}(\dots(\pi_{Enc}(\pi_{Enc}(c_1^*||0^{b_c} \oplus IV) \oplus c_2^*||0^{b_c}) \oplus c_3^*||0^{b_c})\dots \oplus c_e^*||0^{b_c}) \oplus T_1^*||0^{b_c+b_r-r}) \oplus T_2^*||0^{b_c+b_r-r}$

On Encryption-Query $M_d(d \xleftarrow{\$} \{0,1\})$

```
1. $C^f = c_1^*||c_2^*||\dots||c_n^*||T_1^*||T_2^*$
2. $C^e = c_{n+1}^*||\dots||c_e^*$
3. $y = f(C^f)$
4. Return: $y||C^e$
```

On Decryption-Query $y||C^e$
Same as Game G8
On π-Query m, where $m \in \{0,1\}^b$
Same as Game G5
On π^{-1}-Query v. where $v \in \{0,1\}^b$
Same as Game G5

On π_{Enc}-Query m, where $m \in \{0,1\}^b$

```
1. $v = \pi(m)$
2. $I_{Enc} = I_{Enc} \bigcup \{(m,v)\}$
3. return $v$;
```

On $\pi_\mathcal{A}$-Query m, where $m \in \{0,1\}^b$

```
1. if $m = R^*||0^{b_r-r}||IV_2$ $bad \leftarrow true$
2. $v = \pi(m)$
3. $I_\mathcal{A} = I_\mathcal{A} \bigcup \{(m,v)\}$
4. return $v$;
```

On $\pi_\mathcal{A}^{-1}$-Query v, where $v \in \{0,1\}^b$

```
1. $m = \pi^{-1}(v)$
2. if $m = R^*||0^{b_r-r}||IV_2$ $bad \leftarrow true$
3. $I_\mathcal{A} = I_\mathcal{A} \bigcup \{(m,v)\}$
4. return $v$;
```

Figure 15: Game G11

Game G12: Initialise $I_\pi = I_{Enc} = I_{Dec} = I_\mathcal{A} = \emptyset$, $f : \{0,1\}^\ell \to \{0,1\}^\ell$, $IV_1 = 0^{b_i}$ $IV_2 = 0^{b_c}, IV_3 = IV_2 \oplus 1$, $Y = \{IV_2, IV_3\}$, Given $y^*||C^{e*} \overset{\$}{\leftarrow} \{0,1\}^{e*b_i+2r}$, $C^f* = f^{-1}(y^*)$,
Let $C^* = C^f*||C^{e*}$, where $C^f* = c_1^*||c_2^*||\ldots||c_n^*||T_1^*||T_2^*, C^{e*} = c_{n+1}^*||\ldots||c_e^*$
$R^*||* = \pi_{Enc}(\pi_{Enc}(\ldots(\pi_{Enc}(\pi_{Enc}(c_1^*||0^{b_c} \oplus IV) \oplus c_2^*||0^{b_c}) \oplus c_3^*||0^{b_c})\ldots \oplus c_e^*||0^{b_c}) \oplus T_1^*||0^{b_c+b_r-r}) \oplus T_2^*||0^{b_c+b_r-r}$

On Encryption-Query $M_d(d \overset{\$}{\leftarrow} \{0,1\})$

```
1. Return: $y||C^e = f(C^f*)||C^{e*}$;
```

On Decryption-Query $y||C^e$
Same as Game G8

On π-Query m, where $m \in \{0,1\}^b$
Same as Game G5

On π^{-1}-Query v. where $v \in \{0,1\}^b$
Same as Game G5

On π_{Enc}-Query m, where $m \in \{0,1\}^b$

```
1. $v = \pi(m)$
2. $I_{Enc} = 1I_{Enc} \bigcup \{(m,v)\}$
3. return $v$;
```

On $\pi_\mathcal{A}$-Query m, where $m \in \{0,1\}^b$

```
1. $if\ m = R^*||IV_2\ bad \leftarrow true$
2. $v = \pi(m)$
3. $I_\mathcal{A} = I_\mathcal{A} \bigcup \{(m,v)\}$
4. return $v$;
```

On $\pi_\mathcal{A}^{-1}$-Query v, where $v \in \{0,1\}^b$

```
1. $m = \pi^{-1}(v)$
2. $if\ m = R^*||IV_2\ bad \leftarrow true$
3. $I_\mathcal{A} = I_\mathcal{A} \bigcup \{(m,v)\}$
4. return $v$;
```

Red Color text shown hidden/unknown from Game

Figure 16: Game G12

Adversary $\mathcal{B}_\mathcal{A}$: Given $y||C^e \overset{\$}{\leftarrow} \{0,1\}^{e*b_i+2r}$, where $y \in \{0,1\}^\ell$. Find C^f such that $f^{-1}(C^f) = y$.

Adversary $\mathcal{A}$: Initialise $I_\pi = I_{Enc} = I_{Dec} = I_\mathcal{A} = \emptyset$, $f : \{0,1\}^\ell \to \{0,1\}^\ell$, $IV_1 = 0^{b_i}$ $IV_2 = 0^{b_c}, IV_3 = IV_2 \oplus 1$, $Y = \{IV_2, IV_3\}$,

On Encryption-Query $M_d(d \overset{\$}{\leftarrow} \{0,1\})$ by $\mathcal{A}$

```
1. Return: $y||C^e$;
```

On Decryption-Query $y||C^e$ by $\mathcal{A}$
Same as Game G8

On π-Query m, where $m \in \{0,1\}^b$ by $\mathcal{A}$
Same as Game G5

On π^{-1}-Query v. where $v \in \{0,1\}^b$ by $\mathcal{A}$
Same as Game G5

On π_{Enc}-Query m, where $m \in \{0,1\}^b$

```
1. $v = \pi(m)$
2. $I_{Enc} = 1I_{Enc} \bigcup \{(m,v)\}$
3. return $v$;
```

On $\pi_\mathcal{A}$-Query m, where $m \in \{0,1\}^b$

```
1. $v = \pi(m)$
2. $I_\mathcal{A} = I_\mathcal{A} \bigcup \{(m,v)\}$
3. return $v$;
```

On $\pi_\mathcal{A}^{-1}$-Query v, where $v \in \{0,1\}^b$

```
1. $m = \pi^{-1}(v)$
2. $I_\mathcal{A} = I_\mathcal{A} \bigcup \{(m,v)\}$
3. return $v$;
```

Finalization: if $\exists R, T_1, T_2$ such that $((R||0^{b_r-r} \oplus IV_1)||IV_2, v_{1_1}||v_{2_1})$, $(x_e||w_e, T_1||P||v_{2_{e+1}})$, $((z_{1_{e+1}} \oplus T_1||0^{0^{b_r-r}})||u_{2_{e+1}}, (T_2 \oplus R)||P||z_{2_{e+2}}) \in I_\mathcal{A}$ and $f(c_1||\ldots||c_n||T_1||T_2) = y$, then return $C^f = (c_1||\ldots||c_n||T_1||T_2)$, where $P \in \{0,1\}^{b_r-r}$

Figure 17: Adversary $\mathcal{B}_\mathcal{A}$