Strong Leakage Resilient Encryption: Enhancing Data Confidentiality by Hiding Partial Ciphertext [⋆]

Jia Xu[1] and Jianying Zhou[2]

[1] Singtel/Trustwave, Singapore
jiaxu2001@gmail.com
[2] Singapore University of Technology and Design
jianying_zhou@sutd.edu.sg

Abstract. Leakage-resilient encryption is a powerful tool to protect data confidentiality against side channel attacks. In this work, we introduce a new and strong leakage setting to counter backdoor (or Trojan horse) plus covert channel attack, by relaxing the restrictions on leakage. We allow *bounded* leakage at *anytime* and *anywhere* and over *anything*. Our leakage threshold (e.g. 10000 bits) could be much larger than typical secret key (e.g. AES key or RSA private key) size. Under such a strong leakage setting, we propose an efficient encryption scheme which is semantic secure in standard setting (i.e. without leakage) and can tolerate strong continuous leakage. We manage to construct such a secure scheme under strong leakage setting, by hiding partial (e.g. 1%) ciphertext as secure as we hide the secret key using a small amount of more secure hardware resource, so that it is almost equally difficult for any adversary to steal information regarding this well-protected partial ciphertext or the secret key. We remark that, the size of such well-protected small portion of ciphertext is chosen to be much larger than the leakage threshold. We provide concrete and practical examples of such more secure hardware resource for data communication and data storage. Furthermore, we also introduce a new notion of computational entropy, as a sort of computational version of Kolmogorov complexity. Our quantitative analysis shows that, hiding partial ciphertext is a powerful countermeasure, which enables us to achieve higher security level than existing approaches in case of backdoor plus covert channel attacks. We also show the relationship between our new notion of computational entropy and existing relevant concepts, including Shannon-Entropy, Yao-Entropy, Hill-Entropy, All-or-Nothing Transform, and Exposure Resilient Function. This new computation entropy formulation may have independent interests.

Keywords: Leakage Resilient Encryption, Secret Sharing, Information Dispersal Algorithm, Information-theoretic security, Side Channel Attack, Covert Channel Attack, Subliminal channel, Kolmogorov complexity

1 Introduction

Leakage resilient cryptography has been studied for over a decade, aiming to counter side channel attacks, among other goals. Existing works on leakage resilient cryptography typically impose some restrictions on *when*, *where*, or *what* can be leaked. Some work assumes that there exits a leakage-free setup phase. Some works assume there exists a secure hardware device,

[⋆] The first author is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Corporate Laboratory@University Scheme, National University of Singapore, and Singapore Telecommunications Ltd. The second author is supported by the National Research Foundation (NRF), Prime Minister's Office, Singapore, under its National Cybersecurity R&D Programme (Award No. NRF2014NCR-NCR001-31) and administered by the National Cybersecurity R&D Directorate.

such that any computation inside this secure device is leakage-free. If some secret key is stored in such secure device and never leaves from it, then such secret key is assumed to be leakage-free. Some works only allow leakage on secret key. Furthermore, some works consider bounded leakage with a very small upper bound—$O(\mathtt{Poly}(\log \lambda))$ where λ is the security parameter.

1.1 Background in Existing Leakage Models

1.1.1 Bounded Retrieve Model The bounded retrieve model [3,2,14,16] assumes the total amount of leaked information during the lifetime of the attacked system, is upper bounded by a constant ℓ, which could be as large as gigabytes. An existing approach [3,14] is to purposely make the shared secret key size significantly larger than the leakage upper bound—ℓ (e.g. $\geq 2\ell + \lambda$ where λ is the security parameter). In order to make the computation as fast as the case of short secret key, this approach assumes a leakage-free phase, during which, one party (say, Alice) can randomly extract a short session key from the large shared secret key using a random seed. The other party (say, Bob) of communication can re-generate the same short session key from the same shared large secret key after receiving the same random seed.

It is easy to see, under continuous bounded leakage setting, any static secret key can be leaked one bit by one bit, and pseudorandomness technique cannot be applied directly since short seed could be (partially) leaked. Furthermore, we allow $\mathcal{O}(\lambda)$ bits leakage such that leakage threshold could be larger than secret key size (e.g. the short session key in the above paragraph), thus the whole block cipher key (e.g. 128 bits AES key) could be leaked. Therefore, bounded retrieve model does not satisfy our goal.

1.1.2 A leakage-free time period during the computation process of cryptography primitive Alwen, Dodis and Wichs [2] proposed several leakage resilient cryptography primitives with flexible (and possibly very large) key size. A key idea in their authenticated key agreement scheme, is: (1) Generate many keys in the setup; (2) and during a leakage-free time period, the sender and receiver will randomly sample a subset of keys, and use them to authenticate each other; and then establish a short shared session key. As long as a constant fraction of all keys are unknown to the adversary after bounded leakage, a random subset of keys contains at least one unknown key with very high probability. After that, standard cryptography primitives are applied with the short secure session key (e.g. AES).

In our leakage setting, there will be *no* leakage-free time period and any *short* value (e.g. AES key) could be leaked. So we have to seek new approaches.

1.1.3 Secret Key never leaves from Secure Hardware Device The computation power of secure hardware devices (e.g. Trusted Platform Module) may not be able to match the power of desktop Intel/AMD CPU. Secure hardware devices (e.g. Trusted Platform Module [3], and Intel SGX [30]) may also suffer from side channel attacks. Furthermore, there seems no evidence to show that the vendors of secure hardware device are more trusted than vendors of other component (e.g. CPU, GPU, RAM, hard disk, OS, web browser, virtual machine software, etc) in a computer system.

1.1.4 Randomness Extractor One may consider to extract a short block cipher (e.g. AES) key from a long secret key and then encrypt the message using the short block cipher directly. Assuming leakage only occurred before the randomness extractor was applied, (e.g. as the setting of [3,14]), this method will work. But in our setting, we do not make such assumption, and instead we allow bounded leakage at any time.

[3] `https://www.rambus.com/blogs/side-channel-attacks-reportedly-targeted-trusted-platform-modules-2/`

1.1.5 Memoryless Leakage Oracle

An essential difference between leakage oracle in side channel attack in related works and leakage oracle in Trojan horse malware plus covert channel attack in this paper, is that, whether the leakage oracle has cache memory and is allowed to access history data. Some recent works in leakage resilient cryptography [26,8,7] assumes that: (1) for each invocation of cryptography primitive, the leakage threshold is smaller than secret key size; and (2) leakage oracle only takes input from current status of the cryptography computation, and is not allowed to access historical status. They can achieve security by refreshing the secret key frequently (together with other techniques). Imagine a simplified example [26]: To encrypt the i-th message, one may adopt a fresh 256-bit encryption key $k_i := \texttt{SHA256}(k_{i-1})$, and the adversary is allowed to learn only a single bit $\mathcal{L}(k_i) \in \{0,1\}$ over the key k_i. With all leaked information $\{\mathcal{L}(k_j) : j \in [0,i]\}$, a polynomial-time adversary seems not be able to learn some useful knowledge about any secret key. However, in case of Trojan horse plus covert channel attack in this paper, the Trojan horse malware may keep an old key k_0 in a local cache memory, and send out one bit per every invocation of encryption scheme via covert channel. So after encrypting $|k| = 256$ messages, all of 256 bits of k_0 could be sent out to a remote adversary, who can compute every k_i from k_0. With all ciphertexts (which can be obtained via eavesdropping, without resorting to leakage oracle), the adversary can decrypt and recover all plaintexts. Thus 256 bits leakage leads to exposure of everything—all plaintexts and (future) secret keys. Our new security formulation in this work is aiming to prevent such kind of *leakage amplification*.

It will be interesting to study the leakage resilient cryptography with adversary who has limited leakage bandwidth (say ℓ bits per invocation of crypto primitive) and limited cache memory (say w bits memory). In this work, we actually do not assume any upper bound in the size of cache memory. Since covert channel with large bandwidth and/or Trojan horse with large cache memory, may be more easily captured or prevented by existing solution (e.g. anti-virus software and intrusion detection system, Trojan-Resilient hardware [17,10]), it is reasonable to put some small upper bound in values of ℓ and w. We leave this as an open problem.

1.2 Our Contributions

The main contributions of this work can be summarized as below.

1.2.1 New Leakage Setting

Since existing leakage settings does not fit for our goal, we present a new strong leakage model, to capture the threat of backdoor or Trojan horse and covert channels in computer hardware/software systems. We allow *bounded* (e.g. 10000 bits) leakage at *anytime* and *anywhere* and over *anything*, with only two restrictions on the adversary: (1) the adversary algorithms are efficient (probabilistic polynomial time); (2) the bandwidth of the covert channel is bounded from the above. By our knowledge, all existing works designed for leakage settings in Section 1.1 are trivially broken under our leakage setting, since the Trojan horse could observe every step of computation of the victim program (e.g. an encryption program) and then steal the entire short private key. We emphasize that, the white box cryptography [5,20] using program obfuscation, which claims to protect secret key from attackers with direct control of the encryption device, is prohibitively impractical, even for a simple function [13].

In most, if not all, existing works, an adversary is assumed to obtain full information of ciphertext easily (e.g. via eavesdropping) without resorting to any expensive attack methods (e.g. side or covert channel leakage). In contrast, in this paper, we assume that a small portion (e.g. 1%) of ciphertext is as strongly protected as the secret key, so that the adversary has to resort to more advanced method (e.g. backdoor or Trojan horse and covert channel attack) to obtain this portion of ciphertext, just like it steals the secret key. The size of this small portion of ciphertext would be considerably larger than the leakage threshold, which in turn is much larger than the size of the underlying secret key. Later, in Section 2.3, we will support this assumption with real world examples. We compare various settings of encryption schemes in Table 1.

Table 1. Comparison of different encryption settings. Let $\mathtt{Msg} := \mathsf{Decrypt}(k; \mathtt{Ctx}_0 \| \mathtt{Ctx}_1)$ where $|\mathtt{Ctx}_0| \ll |\mathtt{Ctx}_1|$

Formulation	Can adversary access the decryption key k?	Can adversary access ciphertext $(\mathtt{Ctx}_0, \mathtt{Ctx}_1)$	Security Definition	Tools and methodologies
Traditional Encryption (AES, RSA)	No	Full access	Semantic Security (CPA/CCA1/CCA2)	Computationally indistinguishability
Leakage resilient encryption	bounded and *controlled* partial access	Full access	Diverse. Some simple setting may adopt semantic security with leakage oracle query [8,7]	Randomness extractor
This work	bounded but *uncontrolled* partial access	Full access to $\mathtt{Ctx}_1$, and bounded but *uncontrolled* partial access to $\mathtt{Ctx}_0$	Prevent leakage amplification	Secret sharing; variant of Kolmogorov complexity. Computationally indistinguishability is *inappropriate*

1.2.2 Notion of Steal-Entropy We propose a new notion called "steal-entropy", as a sort of computational version of Kolmogorov complexity. With this "steal-entropy", we quantitatively analyse the advantage of our approach over existing works. Our formulation is non-trivial and has to resolve several important issues: (1) Unlike Shannon-Entropy, Yao-Entropy and Hill-Entropy are defined over distribution of random variable, and Kolmogorov complexity is defined over string, our steal-entropy will be defined over an algorithm which converts the distribution of input random variable to the distribution of output random variable. (2) Statistical or computational indistinguishability notion (e.g. semantic security under CPA/CCA/CCA2 attack mode) is inappropriate in our leakage setting, since a single bit of arbitrary leakage will help an adversary to win the guess-game trivially. (3) Kolmogorov complexity is uncomputable in general, but in our formulation, we should avoid to define any uncomputable function. As a result, unlike existing variant formulations of entropy, it is hard to define our steal-entropy as a single scalar value (We will discuss the reason in next section). Instead, we will give an upper bound and a lower bound for the steal-entropy of a given algorithm. To show a program has poor steal-entropy, we need provide a small upper bound on the steal-entropy of this program; to show a program has high steal-entropy, we need provide a large lower bound on the steal-entropy of this program.

1.2.3 Construction We propose an efficient encryption scheme and demonstrate that hiding partial ciphertext could be a powerful tool to defeat strong leakage attack. We construct our encryption scheme using Vandermonde matrix and evaluate the steal-entropy of the proposed scheme without relying on any hard problem assumption. Informally speaking, our encryption scheme will ensure that, without complete ciphertext, the attacker obtains very limited information about the plaintext, even if the attacker has stolen a bounded amount of message (e.g. the entire short private key) of his/her choice. We will compare our solution with some related approaches, including All-or-Nothing Transform and White-Box Cryptography, both of which could not satisfy our goal.

The proposed solution will be used to construct a "virtually isolated network" [34]. We discuss details later in Section 2.

1.3 Organizations

The rest of this paper is organized in this way: Section 2 gives an overview of our work, including our leakage setting, formulation of steal-entropy, and our proposed construction of leakage/steal-resilient encryption scheme. In addition to the related works already discussed in Section 1 and 2, Section 3 discusses more related works. We present our formal formulation of steal-entropy in Section 4, propose and analyse our encryption scheme in Section 5. Before we conclude this paper in Section 6, Section 3 discusses more related works which are not covered in previous sections. Our proofs are given in the appendix.

2 Overview of Our Work

2.1 Our Leakage Setting

2.1.1 Motivation of New Leakage Setting In this paper, we aim to counter not only side channel attack but also covert channel attack. Nowadays, computer systems become so complex and consist of a lot of software/hardware components which are designed, manufactured and sold by various companies from various countries. It is definitely not a trivial task for PC users to check whether some backdoor program or malware (e.g. Trojan horse) has been planted inside his/her PC hardware/software system. The well-known "Dual Elliptic Curve Deterministic Random Bit Generator" (Dual_EC_DRBG) backdoor [4] demonstrates that the potential threat from backdoor in algorithm/software/hardware is not that far away from every computer user. Another serious threat is software Trojans horse or even hardware Trojan horse [5]. The backdoor or Trojan horse malware may observe the victim's computer system to gather information and send collected (possibly compressed) information out via a covert channel or subliminal channel.

Facing such threats from backdoor and Trojan horse, in this work, we have to revise the existing leakage setting: (1) Theoretically, backdoor or Trojan horse programs could be planted by some software/hardware vendor and they exist in victim's computer from the very beginning. So it might not be appropriate to assume a leakage-free time period. (2) Possibly, secure hardware device may be suffering from side channel attack. Even worse, the backdoor program might be planted by vendors of the secure hardware device and the assumption of leakage-free secure hardware device is hard to validate. (3) The backdoor or Trojan horse malware may have their own storage buffers, so history data can be buffered and then leaked 1 bit by 1 bit via the covert channel (thus Pereira, Standaert and Vivek [26] would be broken trivially as discussed in Section 1.1.5).

2.1.2 New Leakage Setting In general, we allow *efficient* leakage with *bounded bandwidth* at *anytime* and *anywhere* and over *anything*. The only two restrictions on leakage are: (1) The

[4] Quotation from `https://en.wikipedia.org/wiki/Kleptography`: "*The Dual_EC_DRBG cryptographically secure pseudo-random number generator from the NIST SP 800-90A is thought to contain a kleptographic backdoor. Dual_EC_DRBG utilizes elliptic curve cryptography, and NSA is thought to hold a private key which, together with bias flaws in Dual_EC_DRBG, allows NSA to decrypt SSL traffic between computers using Dual_EC_DRBG for example.*" Quotation from `https://en.wikipedia.org/wiki/Dual_EC_DRBG`: "*The alleged NSA backdoor would allow the attacker to determine the internal state of the random number generator from looking at the output from a single round (32 bytes); all future output of the random number generator can then easily be calculated, until the CSPRNG is reseeded with an external source of randomness. This makes for example SSL/TLS vulnerable, since the setup of a TLS connection includes the sending of a randomly generated cryptographic nonce in the clear.*"

[5] `http://spectrum.ieee.org/semiconductors/design/stopping-hardware-Trojans-in-their-tracks`

leakage amount of each encryption (i.e. the bandwidth of covert channel) is bounded (e.g. $\mathcal{O}(\lambda)$). In this paper, we are interested in medium value of leakage threshold, e.g. tens of thousands bits, which is much larger than typical private key size (e.g. AES key and RSA private key). (2) The backdoor or Trojan horse program (i.e. the leakage function) is computationally bounded (e.g. polynomial time algorithm). Our setting is closer to study of memory leakage resilient cryptography, and does not follow the assumption that *only computation leaks information* [25].

Recall that, in most, if not all, leakage-resilient cryptography research works, an adversary has two different methods to obtain desired information:

- A *cheap* method to obtain a large amount of weakly protected information, for example, eavesdropping ciphertext on communication link.
- An *expensive* method to obtain a small amount of strongly protected information, for example, using side channel attack or Trojan horse malware plus covert channel attack to obtain partial or full information of the short secret key.

Typically in existing works, an adversary is assumed to obtain full information of ciphertext using the cheap method (e.g. eavesdropping), meanwhile subject to several restrictions on obtaining information of short secret key (e.g. assumed leakage-free time period or hardware device). Unlike existing works, in this paper, we impose minimum restrictions on information leakage, and assume that a small part (e.g. 1% or 0.1%) of ciphertext [6] is as strongly protected as the short secret key, so that the adversary has to resort to the expensive method (e.g. Trojan horse and covert channel) to obtain this part of ciphertext. Next, we will support this assumption with real world examples.

Secure Storage Device. For data storage, we assume there are two categories of storage: one with small capacity is much more secure, but probably more expensive, in term of unit price; the other with large capacity is relatively less secure, but likely cheaper. In case that a user wish to backup large size sensitive historical data in cloud storage server, but did not trust the cloud in data confidentiality. Then this user's local *offline* storage device, which is physically disconnected from any computers and Internet, could be an example of the former, and the cloud storage [7] could be an example of the latter.

Secure Communication Link. For data transmission, we assume there exist two categories of communication channels, one with small bandwidth is much more secure but probably very expensive, such that an adversary cannot obtain the transmitted data with low cost (e.g. eavesdropping); the other with large bandwidth is relatively less secure but likely cheaper, such that an adversary can obtain all transmitted data with low cost. The example of former could be laser communication in free space (e.g via satellites) or even neutrinos communication in the future, which is relatively more difficult to eavesdrop, and the example of latter could be Internet. Another example is "virtually isolated network" [8], recently proposed by Xu and Zhou [34], which is a hybrid network with two communication channels: one is a physically isolated network with small bandwidth, and the other is Internet with large bandwidth. Their work [34] combines these two channels with unidirectional network links (a.k.a data diode or air gap), so that the isolated network will be still always physically isolated from Internet.

[6] The encryption scheme is length-preserving, and the size of ciphertext is equal to the size of plaintext.

[7] Note: (1) Many cloud storage servers provide a certain amount (e.g. 15GB) of free cloud storage for individual users; (2) the cost of offline local storage should include not only hardware purchase cost but also hardware maintenance and storage cost (i.e. keep the harddisk drive in a proper physical environment for a long time).

[8] Actually, the motivation of this work is to provide an extremely secure (*informally, close to physically isolated network*) communication method in this "virtually isolated network" [34]. Here we choose strong leakage resilience against potential backdoor as our formal definition of "extremely secure".

Our strategy is to enhance security level of the large amount of less secure but cheaper hardware resource by leveraging on small amount of more secure but expensive hardware resource, essentially creating a hybrid effects in security. We aim to prevent the adversary from eavesdropping full information of our ciphertext.

2.2 Notion of Steal-Entropy

Unlike previous leakage formulation, we attempt to formalize security in leakage setting from a different angle. We try to answer a very important question:
"At least how many bits should the adversary steal in order to obtain the desired secret information?"

In this work, we are concerning how many bits the adversary has to obtain *using the expensive method*, in order to obtain full or partial information of the plaintext. Informally, we may call this "minimum but sufficient number of leaked/stolen bits" which will lead to compromise of secret plaintext, as the *steal-entropy* of the encryption algorithm.

Let P (e.g encryption algorithm/program) denote the victim algorithm or program. In our formulation, an adversary chooses two algorithms, denoted with steal algorithm S and recovery algorithm R. The steal algorithm S is given oracle access to the whole computation process of P, including any internal states (e.g. secret keys, random seeds, input and any computation steps). Then the steal algorithm S is allowed to pass a short message, which is at most ℓ bits, to the recovery algorithm R, which attempts to output desired secret information. If the recovery algorithm R is able to output the desired secret information with probability close to 1, with value of ℓ much smaller than the size of desired secret information, then we say the victim algorithm P has very low steal-entropy rate. In this work, we are interested in medium value of leakage threshold ℓ (e.g. tens of thousands), which is larger than typical secret key length, but could be much smaller than typical ciphertext length. Figure 1 illustrates our formulation setting. Our notion of "steal-entropy" could be treated as a computation version of Kolmogorov complexity.

2.2.1 Steal-Entropy in Input or Output

Pseudorandom number generators, pseudorandom function and encryption are important cryptography primitives applied to protect data confidentiality. For an algorithm P similar to pseudorandom number generator and pseudorandom function, we are interested to ask a question: Assuming a Trojan horse malware is observing the computation process of algorithm P upon a randomly chosen input x, at least how many bits should the Trojan horse malware steal and send out, in order to allow a remote attacker to recover the output $\mathsf{P}(x)$ of the algorithm P? To address this question, we define a notion called "Steal-Entropy of an algorithm in Output".

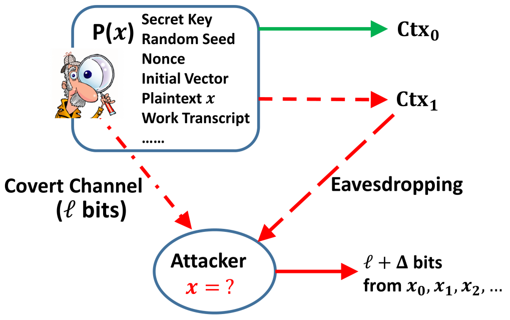
For algorithm P similar to encryption scheme, we are interested to ask another question: Assuming a Trojan horse malware is observing the computation process of algorithm P upon a randomly chosen input x, at least how many bits should this Trojan horse malware steal and send out, in order to allow a remote attacker to recover the input x, where this remote attacker has access to the output [9] $\mathsf{P}(x)$? To address this question, we define a notion called "Steal-Entropy of an algorithm in Input". In addition, to deal with partial information protection, we define a notion called "Strong Steal-Entropy of an algorithm".

2.2.2 A Plausible Formulation of Steal-Entropy

Recall the definition of Shannon-Entropy: The self-information of a string x (sampled from a distribution $\mathcal{X}$), which denotes an event with probability p_x, is defined as $-\log p_x$. The Shannon-Entropy of the distribution $\mathcal{X}$ is

[9] Usually, it is assumed that the adversary has access to the ciphertext.

Fig. 1. Setting of Steal-Entropy and Steal Resilient Encryption. A Trojan horse malware can observe all (secret) information during the execution of algorithm $\mathsf{P}(\cdot)$, including input $x = x_0 \| x_1 \| x_2 \| \ldots$ ($x_i \in \{0,1\}$) , secret keys, random seeds, IV values, ciphertext ($\mathtt{Ctx}_0 \| \mathtt{Ctx}_1$), and all step-by-step work transcript, but is only able to deliver at most ℓ (here we assume $\ell < |x|$) bits message to the remote attacker via some *unidirectional* covert channel. With these ℓ bits stolen-message and whatever he could eavesdrop over Internet, the remote attacker attempts to output $\ell + \Delta$ bits values among x_i's. This paper proposes to "encode" the output of P as two parts: a smaller part $\mathtt{Ctx}_0$ ($|\mathtt{Ctx}_0| > \ell + \lambda$ where λ is the security parameter) and a large part $\mathtt{Ctx}_1$, and transfer or store the small part $\mathtt{Ctx}_0$ using more secure manner, such that the adversary has to resort to advanced technique to steal information about $\mathtt{Ctx}_0$ (e.g. using the Trojan horse virus and the covert channel), rather than eavesdropping. We remark that, (1) we are only interested in length-preserving encryption, such that the bit length of ciphertext (i.e. $|\mathtt{Ctx}_0| + |\mathtt{Ctx}_1|$) is equal to the bit length of plaintext; (2) since $|\mathtt{Ctx}_0| > \ell + \lambda$, so that the attacker could not steal full knowledge of $\mathtt{Ctx}_0$ via the covert channel; (3) typically, $\ell \geq$ SecretKeySize, so the Trojan horse malware may steal the entire secret key. For example, $\ell = 10000$ and SecretKeySize= 128 or 256 for AES, or SecretKeySize ≤ 4096 for typical RSA private key.



defined as the weighted average (or expectation value) $\sum_{x \sim \mathcal{X}} p_x \times (-\log p_x)$ of self-information of x for all x sampled from $\mathcal{X}$.

A plausible formulation for our steal-entropy could be like this: (1) For each input x, we define a function $L(x)$ as the minimum length of leakage message derived by the Trojan horse malware such that the remote attacker is able to recover the output $\mathsf{P}(x)$ from this $L(x)$ bits of leakage-message. (2) Take the expected value (or the average) $\sum_x L(x) \Pr[\mathcal{X} = x]$ as the steal entropy of algorithm P in output.

However, there are several major issues here. (1) Averaging $L(x)$ does not make sense, since we want to introduce an upper bound on $L(x)$, instead of average value of $L(x)$, across different x. An alternative way is that, our steal-entropy could be a range of scalars with upper and lower bound, instead of a single scalar. We remark that we may not simply take the maximum and minimum value of $L(x)$ across all x's. Instead, we introduce a system parameter ϵ (e.g. $\epsilon = 0.05$). We could say the steal-entropy of algorithm P is within range $[a, b]$ with respect to parameter ϵ, if $L(x) \in [a, b]$ for at least $(1 - \epsilon) \times 100\%$ fraction of all possible inputs x. (2) Recall that Kolmogorov complexity is uncomputable in general. Similarly, the function $L(x)$ is likely to be uncomputable, too. So in our real formulation, we need find a way to avoid computing exact value of $L(x)$. Furthermore, just like the formulation of Yao-Entropy and HILL-Entropy, we will provide precise definition for inequality statement "Algorithm P has at least (or at most, respectively) ℓ bits steal-entropy in input (or output, respectively)", rather than defining and computing the exact unique value [10] of steal-entropy.

[10] This unique value could be defined as the integer interval with minimum length satisfying some desired property.

2.2.3 Relation with Existing Similar Notions

We also formally analyze the differences between our notion of steal-entropy with existing similar notions, including Yao-Entropy [35], Hill-Entropy [21], Information Dispersal Algorithm [27], All-or-Nothing Transform [29], and Exposure Resilient Function [11]. We manage to separate our proposed steal-entropy from all of these existing formulations.

2.3 Our Approach

When the leakage threshold ℓ is larger than typical secret key size, most existing encryption schemes and leakage resilient encryption schemes (which only tolerates leakage upto $O(poly \log \lambda) < \lambda$ bits, where λ is the security parameter) would fail to protect data confidentiality, since in typical setting, an adversary could obtain all ciphertext with low cost (e.g. eavesdropping), and the secret decryption key could be stolen by Trojan horse malware and delivered to the remote adversary via covert channel.

Facing such stringent threat of medium size of arbitrary information leakage, two possible directions are: (1) Construct novel encryption scheme with larger flexible key size, say the encryption/decryption key size could be a user-tunable parameter, and range from hundreds bits to hundreds of thousands bits or even more. We will report our work in this direction in a separate paper. We remark that Alwen, Dodis and Wichs [2] does not satisfy our purpose, since this work [2] eventually extracted a short session key from arbitrary large size long term secret key, where this extracted short session key could be stolen under our leakage setting. (2) Break the assumption that the adversary could *easily* obtain all ciphertext. Indeed, this work will attempt to hide a small portion of ciphertext using more secure hardware resource, so that the adversary has to resort to the expensive method to steal information about this small portion of ciphertext, in a similar way that he/she steals the secret key.

2.3.1 Hybrid Idea

In cyber world, hybrid technique is common to achieve some balance between quality and cost: Combining a small amount of high quality and expensive hardware resource with a large amount of low quality and cheap hardware resource, to become a large amount of medium quality hardware. Two typical examples are, (1) in term of price per byte or data read/write speed: CPU cache $\gg$ RAM $\gg$ Hard Disk. An average PC may have a few megabytes of CPU cache, a few gigabytes of RAM, and hundreds or thousands of gigabytes of hard disk. (2) Fusion Drive [11]: combine small size of expensive but faster SSD storage with large size of cheap but slower mechanical hard disk. In fact, in security community, such hybrid idea has been applied for a long time. A typical example is that, the short secret encryption/decryption key is shared among sender and receiver via a much more expensive and more secure way, and large ciphertext is delivered via cheap and insecure way.

2.3.2 Randomness Source

Any static secret information might be stolen one bit by one bit, if backdoor or Trojan horse exists. To defeat continuous leakage/steal with buffer storage, we have to keep investing more and more randomness shared by both sender and receiver. However, it is expensive to generate cryptographically secure shared randomness. In our solution, we will exploit the fact that *plaintext itself is naturally a sort of random source to the view of adversary*, saving the cost to generate true randomness. We protect a small portion of the ciphertext using more secure hardware resource, so that this portion of ciphertext actually acts as another "secret key", which is derived from the plaintext and will change naturally with plaintext, to the view of adversary.

[11] For example, Apple's fusion drive.

2.4 Our Construction

Our leakage setting provides much more freedom and power to adversary, compared to existing works on leakage-resilient cryptography. Consequently, the two very important classical tools, namely *computational indistinguishability* and (statistical or computational) *randomness extractor*, are hardly to be applied under our formulation. In this work, we have to resort to information theory techniques.

Definition 1 (Blockwise Uniform Distribution) *Let $\boldsymbol{y} = (\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n)$, where $\mathbf{y}_i \in \{0,1\}^\rho$ for each $i \in [1,n]$. We say $\boldsymbol{y}$ follows (ζ, ρ)-*Blockwise-Uniform Distribution*, if for any subset $S = \{i_1, i_2, \cdots, i_\zeta\} \subset [1,n]$ with $|S| = \zeta$ and $i_1 < i_2 < i_3 < \cdots < i_\zeta$, we have the joint Shannon-entropy*

$$\mathbb{H}^{\texttt{Shannon}}(\mathbf{y}_{i_1}, \mathbf{y}_{i_2}, \cdots, \mathbf{y}_{i_\zeta}) = \rho\zeta. \tag{1}$$

That is, any subset of ζ distinct blocks $\mathbf{y}_i$ will have joint Shannon entropy equal to their total bit-length (i.e. entropy rate equal to 1).

Remark 1. When $\rho = 1$ and $\zeta = n$, then (ζ, ρ)-Blockwise-Uniform Distribution is identical with uniform distribution.

In this work, we will construct an invertible algorithm P using Vandermonde matrix, such that its inverse algorithm P^{-1}, satisfies this property:

Property 1 *Let $\texttt{Ctx}_0$ and $\texttt{Ctx}_1$ be as in Figure 1, and assume the bit-length $|\texttt{Ctx}_1| = \tau \cdot |\texttt{Ctx0}| = \tau\rho\zeta$. If $\texttt{Ctx}_0$ is independently and uniformly randomly distributed over $\{0,1\}^{\rho\zeta}$, then the output $x = \mathsf{P}^{-1}(\texttt{Ctx}_0, \texttt{Ctx}_1)$ follows (ζ, ρ)-Blockwise-Uniform Distribution, regardless of value of $\texttt{Ctx}_1$ (e.g. this value could be fixed to any given bit-string from its domain).*

Suppose somehow an attacker in Figure 1 is able to output ζ bits among x_i's, say x_{i_j}, $j \in [1, \zeta]$. Then these ζ bits x_{i_j}'s will reside in at most ζ distinct ρ-bit blocks in bit-string x. Since any subset of ζ blocks of x will have Shannon entropy rate equal to 1 (i.e. entropy equal to the bit-length), the collection of these ζ bits x_{i_j}'s will have exactly ζ bits Shannon entropy. Therefore, the adversary has to steal at least ζ bits message via the covert channel, as desired. Apparent, the above proof is not tight with a multiplicative loss of factor ρ. We leaf the tight proof with better security parameters in future work.

3 Related Works

The related works in leakage resilient cryptography have been discussed in Section 1.1. Here we discuss other related works.

Symmetric encryption scheme (e.g. AES, Blowfish [12], and Triple DES [13].) could be the most widely adopted cryptographically secure primitive to protect data confidentiality, especially for large volume of data. AES [12] is a typical example of symmetric encryption scheme, and has been actively adopted in industry and research area due to its security and efficiency for more than one decade.

In additional to encryption techniques, another well-known cryptographic primitive that can be used to protect data confidentiality is "secret-sharing" scheme invented by Shamir [31]. Compared to encryption scheme (e.g. AES [12]) which can only achieve conditional security,

[12] `https://www.schneier.com/academic/blowfish/`

[13] `http://csrc.nist.gov/publications/nistpubs/800-67-Rev1/SP-800-67-Rev1.pdf`

secret-sharing scheme may achieve unconditional security (also known as information-theoretic security), assuming the adversary cannot collect sufficient number of shares.

Despite its strong security, Shamir's secret sharing scheme has significant drawbacks when protecting data confidentiality: (1) for (t, n)-secret sharing scheme, the storage overhead is as large as $(n-1)$ times of size of the secret (i.e. the plaintext to be protected); (2) the reconstruction [24] (or decoding) process is not as efficient as DES or AES.

Rabin [28] proposed "information dispersal algorithm" with zero storage overhead, such that the sum of sizes of all shares is equal to the size of secret message size. His solution is conceptually simple: Let row vector $\boldsymbol{m} = (m_0, m_1, \ldots, m_n)$ be the secret message. Choose an invertible n by n matrix $\mathbf{T}$ with inverse matrix $\mathbf{T}^{-1}$. By multiplying row vector $\boldsymbol{m}$ with matrix $\mathbf{T}$, we obtain the n shares $\boldsymbol{c} = (c_0, c_1, \ldots, c_{n-1}) = \boldsymbol{m} \times \mathbf{T}$. Accordingly, the original secret message $\boldsymbol{m}$ can be recovered by matrix multiplication $\boldsymbol{m} = \boldsymbol{c} \times \mathbf{T}^{-1}$. Othman and Mokdad [9] proposed to protect communication confidentiality by sending each share of message in distinct network path from the same sender to the same receiver.

Alternatively, Krawczyk [23] attempted to make each share shortened, by dividing ciphertext of the long secret message into n pieces, and then apply Shamir's secret sharing scheme over the encryption key. Thus, the storage overhead is linear in short encryption key size and is a fraction of secret message size.

4 Steal-Entropy: How many bits should be stolen to recover the secrete information?

In this section, we propose the notion of "Steal-Entropy". Unlike traditional entropy concepts (e.g. Shannon-Entropy, Yao-Entropy [14], Hill-Entropy, etc) which are defined over random variable with a certain distributions, "steal-entropy" will be defined over algorithms which convert input distribution to output distribution. Our notion of "steal-entropy" could be considered as a computational version of Kolmogorov Complexity [4].

4.1 Background

Definition 2 (Yao-Entropy [35,6,22]) *A distribution $\mathcal{X}$ has **Yao-Entropy** at least ξ, denoted by $\mathbb{H}^{\mathtt{Yao}}_{\epsilon,t}(\mathcal{X}) \geq \xi$, if for every pair of algorithms c and d (called "compressor" and "decompressor") with running time at most t, and $c(\cdot) \in \{0,1\}^{\ell}$,*

$$\Pr_{x \leftarrow \mathcal{X}}[d(c(x)) = x] \leq 2^{\ell - \xi} + \epsilon. \tag{2}$$

Definition 3 (Hill-Entropy [21,6,22]) *A distribution $\mathcal{X}$ has **Hill-Entropy** at least ξ, denoted by $\mathbb{H}^{\mathtt{Hill}}_{\epsilon,t}(\mathcal{X}) \geq \xi$, if there exists a distribution $\mathcal{Y}$ such that $\mathcal{Y}$ has at least ξ bits min-entropy and $\mathsf{D}(\mathcal{X}, \mathcal{Y}) \leq \epsilon$ for any distinguisher program running D in time t.*

Definition 4 (Kolmogorov Complexity [4]) *Kolmogorov Complexity of a string is the length of the shortest possible description of the string in some fixed universal description language (e.g. a program, written in a well-defined programming language, which outputs this string).*

[14] Shannon-Entropy is information-theoretical. Both Yao-Entropy and Hill-Entropy are computational variants.

4.2 Steal-Entropy of an Algorithm in Output

Definition 5 (Steal-Entropy of an Algorithm in Output) *Let* $\mathsf{P} : \{0,1\}^n \to \{0,1\}^m$ *be a deterministic* [15] *single-input algorithm. Let* $\epsilon \in [0, \frac{1}{4})$*. Let* $\mathcal{A}$ *be a t-adversary associated with a pair of algorithms* $(\mathsf{S},\ \mathsf{R})$*, such that*

- *both the steal (or stealage) algorithm* S *and the recovery algorithm* R *are probabilistic algorithms within time t, and*
- *for any non-negative integer* ℓ*, the steal algorithm*

$$\mathsf{S}^{\mathcal{O}(\mathsf{P}(x))}(\ell) \in \{0,1\}^{\leq \ell} \setminus \{\texttt{EmptyString}\}$$

  with oracle access to P*, is allowed to observe all internal states during computation process of algorithm* P *upon an input x, and outputs at most* ℓ *bits non-empty steal-message, and*
- *the recovery algorithm* R *takes as input the steal-message generated by* $\mathsf{S}(\ell)$*, and attempts to guess the value* $\mathsf{P}(x)$*.*

We make the following definitions.

- *We define the advantage of* $\mathcal{A}$ *against* P *w.r.t. input* $x \in \{0,1\}^n$ *as below (before any leakage occurs, x is unknown to* $\mathcal{A}$*)*

$$\mathsf{Adv}^{\mathsf{out}}_{\mathcal{A}(\ell),\mathsf{P}}(x) = \Pr\left[\mathsf{R}\left(\mathsf{S}^{\mathcal{O}\left(y \leftarrow \mathsf{P}(x)\right)}(\ell)\right) = y\right] \tag{3}$$

  where the probability is taken over all random coins of algorithms S *and* R*.*
- *We say the* ***infimum of Steal-Entropy in Output of algorithm*** P *is at least* ξ*, denoted as* $\inf \mathbb{S}^{\mathsf{out}}_{\epsilon,t}(\mathsf{P}) \geq \xi$*, if for any t-adversary* $\mathcal{A}$*, for any non-negative integer* $\ell \leq \xi$*,*

$$\Pr_{x \xleftarrow{R} \{0,1\}^n}\left[\mathsf{Adv}^{\mathsf{out}}_{\mathcal{A}(\ell),\mathsf{P}}(x) \leq \frac{1}{2^{\xi-\ell}} + \epsilon\right] \geq 1 - \epsilon. \tag{4}$$

- *We say the* ***supremum of Steal-Entropy in Output of algorithm*** P *is at most* ξ*, denoted as* $\sup \mathbb{S}^{\mathsf{out}}_{\epsilon,t}(\mathsf{P}) \leq \xi$*, if for some t-adversary* $\mathcal{A}$*,*

$$\Pr_{x \xleftarrow{R} \{0,1\}^n}\left[\mathsf{Adv}^{\mathsf{out}}_{\mathcal{A}(\xi),\mathsf{P}}(x) \geq 1 - \epsilon\right] \geq 1 - \epsilon. \tag{5}$$

- *We say* $\mathbb{S}^{\mathsf{out}}_{\epsilon,t}(\mathsf{P}_0) \geq \mathbb{S}^{\mathsf{out}}_{\epsilon,t}(\mathsf{P}_1)$ *(or equivalently* $\mathbb{S}^{\mathsf{out}}_{\epsilon,t}(\mathsf{P}_1) \leq \mathbb{S}^{\mathsf{out}}_{\epsilon,t}(\mathsf{P}_0)$*), if the following two equations hold*

$$\inf \mathbb{S}^{\mathsf{out}}_{\epsilon,t}(\mathsf{P}_0) \geq \inf \mathbb{S}^{\mathsf{out}}_{\epsilon,t}(\mathsf{P}_1) \tag{6}$$

$$\sup \mathbb{S}^{\mathsf{out}}_{\epsilon,t}(\mathsf{P}_0) \geq \sup \mathbb{S}^{\mathsf{out}}_{\epsilon,t}(\mathsf{P}_1). \tag{7}$$

- *We say* $\mathbb{S}^{\mathsf{out}}_{\epsilon,t}(\mathsf{P}_0) \gg \mathbb{S}^{\mathsf{out}}_{\epsilon,t}(\mathsf{P}_1)$ *(or equivalently,* $\mathbb{S}^{\mathsf{out}}_{\epsilon,t}(\mathsf{P}_1) \ll \mathbb{S}^{\mathsf{out}}_{\epsilon,t}(\mathsf{P}_0)$*), if the following equation holds*

$$\inf \mathbb{S}^{\mathsf{out}}_{\epsilon,t}(\mathsf{P}_0) \geq \sup \mathbb{S}^{\mathsf{out}}_{\epsilon,t}(\mathsf{P}_1). \tag{8}$$

Proposition 1 *Let* $\mathsf{P} : \{0,1\}^n \to \{0,1\}^m$ *be a deterministic algorithm and* $\ell \leq m$ *be a non-negative integer. We have*

[15] When all random coins are treated as a part of input, any probabilistic algorithm will become deterministic.

- $0 \le \inf \mathbb{S}^{\mathtt{out}}_{\epsilon,t}(\mathsf{P}) \le \sup \mathbb{S}^{\mathtt{out}}_{\epsilon,t}(\mathsf{P}) \le \min\{n, m\}$
- $\forall 0 < \epsilon_0 \le \epsilon_1$, $\mathbb{S}^{\mathtt{out}}_{\epsilon_1,t}(\mathsf{P}) \le \mathbb{S}^{\mathtt{out}}_{\epsilon_0,t}(\mathsf{P})$
- $\forall 0 < t_0 \le t_1$, $\mathbb{S}^{\mathtt{out}}_{\epsilon,t_1}(\mathsf{P}) \le \mathbb{S}^{\mathtt{out}}_{\epsilon,t_0}(\mathsf{P})$

Claim 1 *Let* $\mathsf{P} : \{0,1\}^n \to \{0,1\}^n$ *be an identity algorithm such that* $\mathsf{P}(x) = x$ *for each* $x \in \{0,1\}^n$*. then*

- *when* $0 \le \epsilon < 2^{-(n-1)}$, $\sup \mathbb{S}^{\mathtt{out}}_{\epsilon,t}(\mathsf{P}) = n$*;*
- *when* $2^{-(n-1)} \le \epsilon < \frac{1}{4}$, $\sup \mathbb{S}^{\mathtt{out}}_{\epsilon,t}(\mathsf{P}) = n - 1$*;*

(Proof is given in Appendix B.1 on page 23)

4.2.1 Yao-Entropy and Hill-Entropy

Lemma 1 (Steal-Entropy implies Yao-Entropy) *Let* $\mathsf{P} : \{0,1\}^n \to \{0,1\}^m$ *be a deterministic algorithm and* $\xi \le m$ *be a non-negative integer. Let* $\mathcal{X}$ *be a uniform random variable over* $\{0,1\}^n$*. If* $\inf \mathbb{S}^{\mathtt{out}}_{\epsilon,t}(\mathsf{P}) \ge \xi$*, then* $\mathbb{H}^{\mathtt{Yao}}_{2\epsilon,t}(\mathsf{P}(\mathcal{X})) \ge \xi$*. (Proof is given in Appendix B.2 on page 23)*

Lemma 2 (Separation between Yao-Entropy and Steal-Entropy) *Let* $\mathcal{X}$ *be a uniform random variable over* $\{0,1\}^n$*. For any polynomial poly*$(\cdot)$*, there exists a deterministic algorithm* $\mathsf{P} : \{0,1\}^n \to \{0,1\}^m$*, such that*

$$\sup \mathbb{S}^{\mathtt{out}}_{\epsilon,t}(\mathsf{P}) \le n; \qquad \textit{and} \qquad \mathbb{H}^{\mathtt{Yao}}_{\epsilon,t}(\mathsf{P}(\mathcal{X})) \ge \textit{poly}(n). \tag{9}$$

(Proof is given in Appendix B.3 on page 24)

Example 1 *A common practice in cryptography application is that, given a long weak random source (say, x), we apply randomness extractor with seed k to obtain a short almost-uniform random key* $\mathsf{RE}(k;x)$*. Then, we apply pseudorandom function to expand this short but high quality secret key into a long pseudorandom string, which could be longer than x.*

$$\mathsf{P}(k;x) = \mathsf{PRF}_{\mathsf{RE}(k;x)}(0)\|\mathsf{PRF}_{\mathsf{RE}(k;x)}(1)\|\ldots\|\mathsf{PRF}_{\mathsf{RE}(k;x)}(\ell) \tag{10}$$

The Steal-Entropy of this algorithm P *will be at most equal to the length of* $\mathsf{RE}(k;x)$*, and much shorter than both input and output sizes of* P*.*

Lemma 3 (Separation between Hill-Entropy and Steal-Entropy) *Let* $\mathcal{X}$ *be a uniform random variable over* $\{0,1\}^n$*. For any positive valued polynomial poly*$(\cdot)$*, there exists a deterministic algorithm* $\mathsf{P} : \{0,1\}^n \to \{0,1\}^m$*, such that*

$$\sup \mathbb{S}^{\mathtt{out}}_{\epsilon,t}(\mathsf{P}) \le n; \qquad \textit{and} \qquad \mathbb{H}^{\mathtt{Hill}}_{\epsilon,t}(\mathsf{P}(\mathcal{X})) \ge \textit{poly}(n). \tag{11}$$

(Proof is given in Appendix B.4 on page 24)

4.2.2 Exposure Resilient Function and Computational All-or-Nothing Transform

Canetti el al. [11] proposed a concept called "Exposure Resilient Function" (**ERF** for short), and used it to construct computational All-or-Nothing Transforms. Informally, a function $f : \{0,1\}^n \to \{0,1\}^k$ is called a *perfect* (or *statistical* or *computational*) ℓ-**ERF**, if all except ℓ bits of the input x of f is exposed to the adversary, the output $f(x)$ is still *informationally* (or *statistically* or *computationally*) random over $\{0,1\}^k$. Technically, an exposure resilient function can be considered as a deterministic randomness extractor for bit-fixing random source [19]. We quote the Lemma 4.6 in Canetti el al. [11] as below.

Lemma 4 (Lemma 4.6 in Canetti el al. [11]) *Let n, ℓ, m be any polynomially related quantities. Let f be any statistical ℓ-**ERF** (i.e. exposure resilient function) mapping $\{0,1\}^n$ to $\{0,1\}^k$ with negligible statistical deviation ϵ, for some k polynomially related to m. Let G be a pseudo-random generator stretching $\{0,1\}^k$ to $\{0,1\}^m$. Then the function $g(x) = G(f(x)) : \{0,1\}^n \to \{0,1\}^m$ is a computational ℓ-**ERF**.*

Intuitively, in the above lemma, the statistical exposure resilient function f extracts a short high quality randomness $f(x) \in \{0,1\}^k$ from a long but low quality random input x, and then extends the length of output using a standard cryptographically secure pseudorandom generator, such that the result function g will output longer pseudorandomness than the input length, even if all but ℓ bits of the input x is exposed.

Lemma 5 *For any positive-valued polynomial $poly(\cdot)$ and any positive integer k, there exits a polynomial time computable function $\mathsf{P} : \{0,1\}^n \to \{0,1\}^m$, such that*

- P *can be resilient to as large as $poly(k)$ bits leakage under the formulation of* exposure resilient function, *precisely,* P *is a computational ℓ-**ERF** with $\ell = n - poly(k)$;*
- P *can be only resilient to as small as k bits leakage under the formulation of this paper, precisely, the steal-entropy in output is* $\sup \mathbb{S}^{\mathtt{out}}_{\epsilon,t}(\mathsf{P}) \le k$

(Proof is given in Appendix B.5 on page 24)

4.3 Steal-Entropy of an Algorithm in Input

Definition 6 (Steal-Entropy of an Algorithm in Input) *Let $\mathsf{P} : \{0,1\}^n \to \{0,1\}^m$ be a deterministic* [16] *single-input algorithm. Let $\epsilon \in [0, \frac{1}{4})$. Let $\mathcal{A}$ be a t-adversary associated with a pair of algorithms $(\mathsf{S},\ \mathsf{R})$, such that*

- *both the steal (or stealage) algorithm S and the recovery algorithm R are probabilistic algorithms within time t, and*
- *for any non-negative integer ℓ, the steal algorithm*

$$\mathsf{S}^{\mathcal{O}(\mathsf{P}(x))}(\ell) \in \{0,1\}^{\le \ell} \setminus \{\texttt{EmptyString}\}$$

  with oracle access to P, is allowed to observe all internal states during computation process of algorithm P upon an input x, and outputs at most ℓ bits non-empty steal-message, and
- *the recovery algorithm R takes as input the value $\mathsf{P}(x)$ and the steal-message generated by $\mathsf{S}(\ell)$, and attempts to guess the value x.*

We make the following definitions.

- *We define the advantage of $\mathcal{A}$ against P w.r.t. input $x \in \{0,1\}^n$ as below*

$$\mathsf{Adv}^{\mathtt{in}}_{\mathcal{A}(\ell),\mathsf{P}}(x) = \Pr\left[\mathsf{R}\left(\mathsf{S}^{\mathcal{O}\left(y \leftarrow \mathsf{P}(x)\right)}(\ell),\ y\right) = x\right] \tag{12}$$

  where the probability is taken over all random coins of algorithms S and R.
- *We say the **infimum of Steal-Entropy in Input of algorithm** P is at least ξ, denoted as $\inf \mathbb{S}^{\mathtt{in}}_{\epsilon,t}(\mathsf{P}) \ge \xi$, if for any t-adversary $\mathcal{A}$, for any non-negative integer $\ell \le \xi$,*

$$\Pr_{x \xleftarrow{R} \{0,1\}^n}\left[\mathsf{Adv}^{\mathtt{in}}_{\mathcal{A}(\ell),\mathsf{P}}(x) \le \frac{1}{2^{\xi-\ell}} + \epsilon\right] \ge 1 - \epsilon. \tag{13}$$

[16] When all random coins are treated as a part of input, any probabilistic algorithm will become deterministic.

- *We say the* ***supremum of Steal-Entropy in Input of algorithm*** P *is at most* ξ*, denoted as* $\sup \mathbb{S}^{\mathtt{in}}_{\epsilon,t}(\mathsf{P}) \le \xi$*, if for some* t*-adversary* $\mathcal{A}$*,*

$$\Pr_{x \xleftarrow{R} \{0,1\}^n} \left[\mathsf{Adv}^{\mathtt{in}}_{\mathcal{A}(\xi),\mathsf{P}}(x) \ge 1-\epsilon \right] \ge 1-\epsilon. \tag{14}$$

- *We say* $\mathbb{S}^{\mathtt{in}}_{\epsilon,t}(\mathsf{P}_0) \ge \mathbb{S}^{\mathtt{in}}_{\epsilon,t}(\mathsf{P}_1)$ *(or equivalently* $\mathbb{S}^{\mathtt{in}}_{\epsilon,t}(\mathsf{P}_1) \le \mathbb{S}^{\mathtt{in}}_{\epsilon,t}(\mathsf{P}_0)$*), if the following two equations hold*

$$\inf \mathbb{S}^{\mathtt{in}}_{\epsilon,t}(\mathsf{P}_0) \ge \inf \mathbb{S}^{\mathtt{in}}_{\epsilon,t}(\mathsf{P}_1); \qquad \sup \mathbb{S}^{\mathtt{in}}_{\epsilon,t}(\mathsf{P}_0) \ge \sup \mathbb{S}^{\mathtt{in}}_{\epsilon,t}(\mathsf{P}_1). \tag{15}$$

- *We say* $\mathbb{S}^{\mathtt{in}}_{\epsilon,t}(\mathsf{P}_0) \gg \mathbb{S}^{\mathtt{in}}_{\epsilon,t}(\mathsf{P}_1)$ *(or equivalently,* $\mathbb{S}^{\mathtt{in}}_{\epsilon,t}(\mathsf{P}_1) \ll \mathbb{S}^{\mathtt{in}}_{\epsilon,t}(\mathsf{P}_0)$*), if the following equation holds*

$$\inf \mathbb{S}^{\mathtt{in}}_{\epsilon,t}(\mathsf{P}_0) \ge \sup \mathbb{S}^{\mathtt{in}}_{\epsilon,t}(\mathsf{P}_1). \tag{16}$$

Proposition 2 *If* P *is an invertible algorithm, and the inverse algorithm* P^{-1} *has running time* $\le t$*, then* $\inf \mathbb{S}^{\mathtt{in}}_{\epsilon,t}(\mathsf{P}) = \sup \mathbb{S}^{\mathtt{in}}_{\epsilon,t}(\mathsf{P}) = 0$*.*

When the encryption/decryption key is fixed, an encryption algorithm Enc is an invertible algorithm from plaintext to ciphertext. Before any information leakage, an adversary may have knowledge of the whole family $\{\mathsf{Enc}_k\}_{k \leftarrow \mathsf{KGen}(1^\lambda)}$ and do not know which one is picked from this family of permutation algorithms. By stealing the key k, an adversary is able to recover plaintext from ciphertext. This simple fact is summarized as below.

Proposition 3 *For any PPT encryption scheme* $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *and for any key* k *generated by* KGen*, we have*

$$\sup \mathbb{S}^{\mathtt{in}}_{\epsilon,t}\big(\mathsf{Enc}_k\big) \le |k|, \ \textit{where } \epsilon = 0, \ \textit{and } t = \textit{poly}(\cdot). \tag{17}$$

4.4 Discussion

An interesting question is to evaluate the steal-entropy for classical hard problems: factorization problem and discrete log problem, where thousands (say 2048) bits long key provides roughly 80 bits security level. $\mathsf{P}_{\mathtt{Fact}}(p,q) = p \times q$ where both p and q are primes with equal bit-length. $\mathsf{P}_{\mathtt{Log}}(x) = g^x \mod p$ where both g and p are public constants, p is a prime and g is a generator modulo p. Will the steal-entropy of these algorithm be closer to their key size (i.e. thousands) or security level (i.e. 80)? We leave it as an open problem.

4.5 Strong Steal-Entropy in Input

Informally, after stealing ℓ bits arbitrary message, the adversary should be unable to output $\ell + \Delta$ bits information about the secret value, and the amount of leakage amplification is at most Δ bits.

Definition 7 (Strong Steal-Entropy of an Algorithm in Input) *Let* $\mathsf{P} : \{0,1\}^n \to \{0,1\}^m$ *be a deterministic* [17] *single-input algorithm. Let* $\epsilon \in [0, \frac{1}{4})$*. Let* $\mathcal{A}$ *be a* t*-adversary associated with a pair of algorithms* $(\mathsf{S},\ \mathsf{R})$*, such that*

- *both the steal (or stealage) algorithm* S *and the recovery algorithm* R *are probabilistic algorithms within time* t*, and*

[17] When all random coins are treated as a part of input, any probabilistic algorithm will become deterministic.

- *for any non-negative integer ℓ, the steal algorithm*

$$\mathsf{S}^{\mathcal{O}(\mathsf{P}(x))}(\ell) \in \{0,1\}^{\leq \ell} \setminus \{\texttt{EmptyString}\}$$

  with oracle access to P, is allowed to observe all internal states during computation process of algorithm P upon an input x, and outputs at most ℓ bits non-empty steal-message, and
- *the recovery algorithm R takes 2 inputs: (1) the steal-message generated by $\mathsf{S}(\ell)$, and (2) the value $\mathsf{P}(x)$, and outputs two values: (1) $\bar{x} \in \{0,1\}^n$, which is a guess of x, and (2) a subset of indices $\mathbf{I}_x \subset [1,n]$.*

We introduce the following definitions.

- *For any adversary $\mathcal{A}$ with steal algorithm S and recovery algorithm R, let us define the set $\mathbf{G}_{\texttt{msg}}$ of good steal-message as below*

$$\mathbf{G}^{\mathsf{R}}_{\texttt{msg}}(\ell, \Delta, x, \beta) \overset{\text{def}}{=} \left\{ \texttt{Msg} \in \{0,1\}^{\leq \ell} : \begin{array}{l} (\bar{x}, \mathbf{I}) \leftarrow \mathsf{R}(\texttt{Msg}, \mathsf{P}(x)); \\ |\mathbf{I}| \geq \ell + \Delta; \\ \forall i \in \mathbf{I}, \Pr[\bar{x}[i] = x[i]] \geq \beta \end{array} \right\} \tag{18}$$

  where the probability is taken over the random coins of R.
- *Similarly, let us define the set $\mathbf{G}_x$ of good input x as below*

$$\mathbf{G}^{\mathsf{S},\mathsf{R}}_{\texttt{x}}(\ell, \Delta, \alpha, \beta) \overset{\text{def}}{=} \left\{ x \in \{0,1\}^n : \Pr[\mathsf{S}^{\mathcal{O}(\mathsf{P}(x))}(\cdot) \in \mathbf{G}^{\mathsf{R}}_{\texttt{msg}}(\ell, \Delta, x, \beta)] \geq \alpha \right\} \tag{19}$$

  where the probability is taken over the random coins of S.
- *We say the* ***supremum of Strong Steal-Entropy in Input of algorithm*** *P is at most ξ, denoted as $\sup \mathbb{S}^{\texttt{sin}}_{\epsilon,t}(\mathsf{P}) \leq \xi$, if for some t-adversary $\mathcal{A} = (\mathsf{S}, \mathsf{R})$,*

$$\Pr_{x \in_R \{0,1\}^n}[x \in \mathbf{G}^{\mathsf{S},\mathsf{R}}_{\texttt{x}}(\xi, \varsigma(\xi, \epsilon) + 1 - \ell, 1 - \epsilon, 1 - \epsilon)] \geq 1 - \epsilon \tag{20}$$

  where function $\varsigma(\cdot, \cdot)$ is defined as below [18]

$$\varsigma(\ell, \epsilon) \overset{\text{def}}{=} \begin{cases} \ell, & \text{if } 0 \leq \epsilon < 2^{-(\ell-1)} \\ \ell + 1, & \text{if } 2^{-(\ell-1)} \leq \epsilon < \frac{1}{4}. \end{cases} \tag{21}$$

- *Let $\epsilon \geq \lambda^{-c}$ where c could be any positive integer. We say the* ***infimum of Strong Steal-Entropy in Input of algorithm*** *P is at least ξ, denoted as $\inf \mathbb{S}^{\texttt{sin}}_{\epsilon,t}(\mathsf{P}) \geq \xi$, if for any t-adversary $\mathcal{A} = (\mathsf{S}, \mathsf{R})$, for any ℓ with $\varsigma(\ell, \epsilon) = \ell + 1 < \xi$,*

$$\Pr_{x \in_R \{0,1\}^n}[x \in \mathbf{G}^{\mathsf{S},\mathsf{R}}_{\texttt{x}}(\ell, \varsigma(\ell, \epsilon) + 1 - \ell, 0.5 + \epsilon, 0.5 + \epsilon)] \leq 0.5 + negl(\lambda), \tag{22}$$

  where λ is the security parameter, and $negl(\cdot)$ denotes some negligible function.
- *We say $\mathbb{S}^{\texttt{sin}}_{\epsilon,t}(\mathsf{P}_0) \geq \mathbb{S}^{\texttt{sin}}_{\epsilon,t}(\mathsf{P}_1)$ (or equivalently $\mathbb{S}^{\texttt{sin}}_{\epsilon,t}(\mathsf{P}_1) \leq \mathbb{S}^{\texttt{sin}}_{\epsilon,t}(\mathsf{P}_0)$), if the following two equations hold*

$$\inf \mathbb{S}^{\texttt{sin}}_{\epsilon,t}(\mathsf{P}_0) \geq \inf \mathbb{S}^{\texttt{sin}}_{\epsilon,t}(\mathsf{P}_1); \qquad \sup \mathbb{S}^{\texttt{sin}}_{\epsilon,t}(\mathsf{P}_0) \geq \sup \mathbb{S}^{\texttt{sin}}_{\epsilon,t}(\mathsf{P}_1). \tag{23}$$

[18] The reason behind the definition of $\varsigma(\ell, \sigma)$ (i.e. Equation 21) is in our proof of Claim 1. Informally speaking, some steal algorithm $\mathsf{S}(\ell)$ is able to convey *almost* $\ell + 1$ bits message to R algorithm, since $|\{0,1\}^{\leq \ell}| \approx |\{0,1\}^{\ell+1}|$. When the error bound $\epsilon \geq 2^{-(\ell-1)}$, we do not care the difference between such "almost" $\ell + 1$ bits message and actual $\ell + 1$ bits message.

- *We say* $\mathbb{S}^{\mathtt{sin}}_{\epsilon,t}(\mathsf{P}_0) \gg \mathbb{S}^{\mathtt{sin}}_{\epsilon,t}(\mathsf{P}_1)$ *(or equivalently,* $\mathbb{S}^{\mathtt{sin}}_{\epsilon,t}(\mathsf{P}_1) \ll \mathbb{S}^{\mathtt{sin}}_{\epsilon,t}(\mathsf{P}_0)$*), if the following equation holds*

$$\inf \mathbb{S}^{\mathtt{sin}}_{\epsilon,t}(\mathsf{P}_0) \geq \sup \mathbb{S}^{\mathtt{sin}}_{\epsilon,t}(\mathsf{P}_1). \tag{24}$$

Lemma 6 (Amplification) *If there exists some t-adversary $\mathcal{A}_0 = (\mathsf{S}_0, \mathsf{R}_0)$, such that for any positive integer c, and for any $\epsilon \geq \lambda^{-c}$, we have*

$$\Pr_{x \in_R \{0,1\}^n}[x \in \mathbf{G}^{\mathsf{S}_0,\mathsf{R}_0}_{\mathtt{x}}(\ell, \varsigma(\ell,\epsilon) + 1 - \ell, 0.5 + \epsilon, 0.5 + \epsilon)] \geq \mu \tag{25}$$

then there exists some $t \cdot \Theta(1/\epsilon)$-adversary $\mathcal{A}_1 = (\mathsf{S}_1, \mathsf{R}_1)$, such that

$$\Pr_{x \in_R \{0,1\}^n}[x \in \mathbf{G}^{\mathsf{S}_1,\mathsf{R}_1}_{\mathtt{x}}(\ell, \varsigma(\ell,\epsilon) + 1 - \ell, 1 - negl(\lambda), 1 - negl(\lambda))] \geq \mu \tag{26}$$

where λ is the security parameter and $negl(\cdot)$ denotes some negligible function. (The proof is given in Appendix B.6 on page 24)

Definition 8 (Strong Steal-Entropy Rate in Input) *Let $\mathsf{P} : \{0,1\}^n \to \{0,1\}^m$ be a deterministic single-input algorithm. We define the infimum and supremum of steal-entropy rate of algorithm P as*

$$\mu^{\perp} \stackrel{\text{def}}{=} \frac{\inf \mathbb{S}^{\mathtt{sin}}_{\epsilon,t}(\mathsf{P})}{n}; \qquad \mu^{\top} \stackrel{\text{def}}{=} \frac{\sup \mathbb{S}^{\mathtt{sin}}_{\epsilon,t}(\mathsf{P})}{n} \tag{27}$$

(Note that this is a counterpart notion of "entropy rate" or "leakage rate".)

Theorem 7 (Separation between Steal-Entropy and Strong Steal-Entropy) *There exists a constant $c > 0$, such that for any positive integer N, we can construct an algorithm P, such that* $\sup \mathbb{S}^{\mathtt{sin}}_{\epsilon,t}(\mathsf{P}) \leq c$ *and* $\inf \mathbb{S}^{\mathtt{in}}_{\epsilon,t}(\mathsf{P}) \geq N$*. (Proof is given in Appendix B.7)*

4.5.1 All-or-Nothing Transform: Rivest's Package Transform

To be self-contained, we *quote* the All-or-Nothing Transform, called "Package Transform", proposed by Rivest [29] in Appendix A.1 on page 22.

On one side, our approach is similar to All-or-Nothing Transform proposed by Rivest [29], in the sense that we also hide a small portion of ciphertext. Without full knowledge of all ciphertext, it is hard to understand the plaintext. However, on the other hand, an essential difference between our approach and All-or-Nothing Transform (e.g [29]) is that our formulation allows leakage of any $\leq \ell$ bits (possibly aggregated) message, and the value of ℓ could be larger than secret key size. Under such strong leakage setting, the above Package Transform method by Rivest [29] is trivially vulnerable.

Lemma 8 *Package Transform algorithm by Rivest [29] has very small strong steal-entropy in input. Precisely, let PkgTr denote the Package Transform algorithm, we have:*

- $\sup \mathbb{S}^{\mathtt{sin}}_{\epsilon,t}(\mathsf{PkgTr}) \leq |K'|$
- *The strong steal-entropy rate in input of the Package Transform, defined as* $\mu^{\top} \stackrel{\text{def}}{=} \frac{\sup \mathbb{S}^{\mathtt{sin}}_{\epsilon,t}(\mathsf{PkgTr})}{n} = 1/\Theta(n)$ *is approaching to zero, when the input size n approaching to infinity.*

(Proof is given in Appendix B.8 on page 25)

5 Our Proposed Encryption (or Encoding) Scheme

We will describe our proposed encryption scheme in two steps following a modular design.

5.1 Our Steal-Resilient Encryption (or Encoding) Scheme

Definition 9 (Steal-Resilient Encryption/Encoding) *Let $\Phi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a* length-preserving *encryption scheme. Let algorithm* $\textsc{Suffix}_\Phi$ *be defined as below*

$$\textsc{Suffix}_\Phi(k; x) = C_1, \text{ where } k := \mathsf{KeyGen}(1^\lambda) \\ \text{and } C_0 \| C_1 := \mathsf{Encrypt}(k; x) \text{ and } |C_1| = \tau |C_0|. \tag{28}$$

Let n denote the length of plaintext. We say Φ is a $\delta(n)$-steal-resilient encryption scheme with split-factor τ, if the algorithm $\textsc{Suffix}_\Phi$ has infimum of strong steal-entropy rate $\mu^\perp = \frac{\inf \mathbb{S}^{\mathtt{sin}}_{\epsilon,t}(\textsc{Suffix}_\Phi)}{n} \geq \delta(n)$, where $\delta(n) \in [0,1]$ with 1 meaning the best and 0 meaning the worst, $t = O(poly(\lambda))$, and $\epsilon \geq \lambda^{-c}$ for some positive integer c.

We remark that, under our definition, most existing encryption schemes (including any existing block cipher under any existing mode of operation, and All-or-Nothing Transform by Rivest [29], and Leakage resilient encryption [19] [26,2,18,32,1,15,36]) are poorly $\delta(n)$-steal resilient encryption with $\delta(n) = 1/\Theta(n)$ approaching to zero when n approaches to infinity.

We found that the linear transformation with Vandermonde matrix is a good steal-resilient encryption scheme. Let ρ be some positive integer (e.g. 8 or 16 or 32) and $GF(2^\rho)$ be a finite field with order 2^ρ.

We construct an encryption scheme $\Phi_0 = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ as below.

$\Phi_0.\mathsf{KeyGen}(1^\lambda) \to \mathbf{M}$

1. Randomly choose a $\zeta \cdot (1+\tau)$ by $\zeta \cdot (1+\tau)$ Vandermonde matrix [20], and denote its transpose matrix as $\mathbf{M} = (M_{i,j})_{i,j \in [1, \zeta\cdot(1+\tau)]}$, where $M_{i,j} = \alpha_j^i \in GF(2^\rho) \setminus \{0\}$. The inverse of matrix $\mathbf{M}$ exists and is denoted as $\mathbf{M}^{-1}$.
2. Output $\mathbf{M}$.

$\Phi_0.\mathsf{Encrypt}(\mathbf{M}; \boldsymbol{x})$, where $\mathbf{M}$ is a $\zeta \cdot (1+\tau)$ by $\zeta \cdot (1+\tau)$ matrix and $\boldsymbol{x} \in GF(2^\rho)^{\zeta\cdot(1+\tau)}$ is a row vector of dimension $\zeta \cdot (1+\tau)$ (equivalently, 1 by $\zeta \cdot (1+\tau)$ matrix)

1. Compute product $\boldsymbol{y} := \boldsymbol{x} \times \mathbf{M}^{-1}$ of two matrix $\boldsymbol{x}$ and $\mathbf{M}^{-1}$.
2. Treat $\boldsymbol{y}$ as a bit string with length $(1+\tau)\rho\zeta$ bits, which is the concatenation of $\zeta(1+\tau)$ number of ordered ρ-bits finite field elements.
3. Let $\boldsymbol{y}_0$ be the prefix of $\boldsymbol{y}$ with length equal to $\rho\zeta$ bits.
4. Let $\boldsymbol{y}_1$ be the suffix of $\boldsymbol{y}$ with length equal to $\tau\rho\zeta$ bits.
5. Output $(\boldsymbol{y}_0, \boldsymbol{y}_1)$.

$\Phi_0.\mathsf{Decrypt}(\mathbf{M}; \boldsymbol{y}_0, \boldsymbol{y}_1)$

1. Let $\boldsymbol{y}$ be the concatenation of $\boldsymbol{y}_0$ and $\boldsymbol{y}_1$.
2. Parse bit-string $\boldsymbol{y}$ as a row vector of dimension $\zeta(1+\tau)$ where each vector element is from $GF(2^\rho)$.
3. Compute matrix product $\boldsymbol{x} := \boldsymbol{y} \times \mathbf{M}$.
4. Output $\boldsymbol{x}$.

[19] We remark that some of these cited leakage resilient cryptography works actually propose leakage resilient pseudorandom generator/functions, instead of an encryption scheme. These pseudorandom generator/functions can be converted into encryption scheme using classical methods. These resulting encryption schemes will be a poor steal-resilient encryption.

[20] The matrix row/column index starts with either zero or one, makes no essential difference to the property of Vandermonde matrix.

We remark that, any linear transformation with an invertible matrix could constitute an information dispersal algorithm [27], but is unlikely a steal-resilient encryption.

Our experiments in a Macbook Pro Laptop with Intel i5 CPU (purchased in 2014) show that the encryption or decryption can be done in 0.037 seconds (about 21 megabytes per second) with a single CPU core when dimension of $\mathbf{M}$ is 12800 and $\rho = 16, \tau = 31$; and in 0.149 seconds when dimension is 25600 and $\rho = 16, \tau = 63$.

Theorem 9 *Let $\boldsymbol{x} := \boldsymbol{y} \times \mathbf{M}$ be as stated in the above scheme. Then $\boldsymbol{x}$ follows (ζ, ρ)-Blockwise-Uniform distribution, as defined in Definition 1 on page 10. More precisely, parse $\boldsymbol{x}$ as a sequence of elements $(x_1, x_2, \cdots, x_i, \cdots, x_{\zeta(1+\tau)})$ with each element $x_i \in GF(2^\rho)$. If the last $\tau \cdot \zeta$ elements of $\boldsymbol{y}$ is given and fixed, and the first ζ elements of $\boldsymbol{y}$ uniformly distributes over $\{0,1\}^{\rho\zeta}$, then any tuple of ζ elements $(\cdots, x_{i_j}, \cdots)_{j\in[1,\zeta]}$, with distinct indices i_j's, will have exactly $\rho \cdot \zeta$ bits Shannon-Entropy (i.e. the Shannon-Entropy rate is 1).*

Proof. Since $\boldsymbol{x} := \boldsymbol{y} \times \mathbf{M}$, we have

$$x_i = \langle \boldsymbol{y},\ \boldsymbol{M}_i \rangle \in GF(2^\rho), \forall i \in [1, (1+\tau)\zeta] \left(\cdots x_{i_j} \cdots\right)_{j\in[1,\zeta]} = \boldsymbol{y} \times \left(\cdots \boldsymbol{M}_{i_j} \cdots\right)_{j\in[1,\zeta]} \quad (29)$$

where $\boldsymbol{M}_i$ denotes the column vector of the i-th column of matrix $\mathbf{M}$. Furthermore, we can derive

$$\left(\cdots x_{i_j} \cdots\right)_{j\in[1,\zeta]} = \textsc{Prefix}(\boldsymbol{y}, \zeta) \times \left(\cdots \textsc{Prefix}\left(\boldsymbol{M}_{i_j}, \zeta\right) \cdots\right)_{j\in[1,\zeta]} + \boldsymbol{z}, \quad (30)$$

where $\textsc{Prefix}(\boldsymbol{y}, \zeta)$ (respectively,$\textsc{Prefix}(\boldsymbol{M}_{i_j}, \zeta)$) denotes the vector of the first ζ elements from $\boldsymbol{y}$ (respectively, $\boldsymbol{M}_{i_j}$), and $\boldsymbol{z}$ is some constant vector. Since $\boldsymbol{M}$ is the transpose of Vandermonde matrix, the resulting matrix $\left(\cdots \textsc{Prefix}\left(\boldsymbol{M}_{i_j}, \zeta\right) \cdots\right)_{j\in[1,\zeta]}$ will also be the transpose of another Vandermonde matrix. Note that $\textsc{Prefix}(\boldsymbol{y}, \zeta)$ is uniformly distributed over $\{0,1\}^{\rho\zeta}$, due to property of Vandermonde matrix, it is straightforward that the left hand side $\left(\cdots x_{i_j} \cdots\right)_{j\in[1,\zeta]}$ of Equation 30 is uniformly distributed over $\{0,1\}^{\rho\zeta}$, as desired.

Corollary 10 *The proposed scheme Φ_0 is a $\delta(n)$-steal-resilient encryption, with $\delta(n) = \frac{1}{\rho(\tau+1)}$ independent on plaintext length $n = \rho\zeta(1+\tau)$, and $\inf \mathbb{S}^{\mathtt{sin}}_{\epsilon,t}(\textsc{Suffix}_{\Phi_0}) \geq \zeta$. We remark that both ρ and τ are system parameters independent on plaintext length n. (Proof is given in Appendix B.9 on page 26)*

We observe that, in the proof of Theorem 9, we only require the first ζ rows of matrix $\mathbf{M}$ satisfy the special Vandermonde matrix property. Therefore, we could simply tweak the rest rows of matrix $\mathbf{M}$, in order to speed up the decryption performance.

Corollary 11 *In algorithm Φ_0.KeyGen, change the last $\tau\zeta$ rows of matrix $\mathbf{M}$ to a sparse matrix, such that $\mathbf{M}$ is still invertible. Then the resulting variant version of Φ_0 is still $\delta(n)$-steal-resilient encryption, with $\delta(n) = \frac{1}{\rho(\tau+1)}$, where $n = \rho\zeta(1+\tau)$.*

The above Corollary 11 actually separates our notion from secret-sharing scheme: After the tweak in the above corollary, the resulting scheme is no longer a secret sharing scheme.

5.2 Combine Steal-Resilient Encryption and Semantic Secure Encryption

We wish to combine both of the advantage of Steal-Resilient Encryption in leakage setting, and the advantage of semantic secure encryption in standard adaptive chosen message/plaintext attack setting (CCA2/CPA2).

Let Φ_0 be the steal-resilient encryption scheme defined above. Let Φ_1 be a given semantic-secure encryption scheme (precisely, CTR mode of a semantic secure block cipher). Eventually, our encryption scheme Φ_2 is defined as below

- $\Phi_2.\mathsf{KeyGen}(1^\lambda) \leftarrow (k, k_0, k_1)$:
  1. Compute key $\mathbf{M} \leftarrow \Phi_0.\mathsf{KeyGen}(1^\lambda)$.
  2. Compute key $k \leftarrow \Phi_1.\mathsf{KeyGen}(1^\lambda)$.
  3. Output $(k, \mathbf{M})$.

- $\Phi_2.\mathsf{Encrypt}(k, \mathbf{M}; \mathtt{Msg}) \to (\mathtt{C}_0, \mathtt{C}_1)$
  1. Encrypt plaintext Msg using semantic secure encryption to obtain ciphertext $\mathtt{Ctx} \leftarrow \Phi_1.\mathsf{Encrypt}(k; \mathtt{Msg})$.
  2. Split the ciphertext Ctx into two shares using steal-resilient encryption $(C_0, C_1) \leftarrow \Phi_0.\mathsf{Encrypt}(\mathbf{M}; \mathtt{Ctx})$.
  3. Output (C_0, C_1).

- $\Phi_2.\mathsf{Dec}(k, \mathbf{M}; C_0, C_1)$
  1. Merge the two shares C_0 and C_1 as ciphertext $\mathtt{Ctx} \leftarrow \Phi_0.\mathsf{Decrypt}(\mathbf{M}; C_0, C_1)$.
  2. Decrypt Ctx as $\mathtt{Msg} \leftarrow \Phi_1.\mathsf{Decrypt}(k; \mathtt{Ctx})$.
  3. Output Msg.

We remark that, in our proposed scheme, for large input size, Φ_1 can run in CTR mode and Φ_0 can run over every $\rho\zeta(1+\tau)$-bit segment in ciphertext of Φ_1 independently.

Theorem 12 *Let Φ_2 be the proposed encryption scheme by combining a steal-resilient encryption Φ_0 and a semantic secure encryption Φ_1. Then Φ_2 is semantic-secure in standard model, and is $\delta(n)$-steal-resilient encryption with split-factor τ in our leakage-model, where $1/\delta(n) = \rho(\tau+1) + O(1)$.*

Proof (Sketch Proof). The semantic security of Φ_2 is simply implied by the semantic security of Φ_1, we omit the details. Since the result of blockwise uniform distribution XOR another independent distribution (i.e. the pseudorandom bit-sequence generated using the CTR mode of block cipher Φ_1) is still blockwise uniform distribution, Φ_0 is $\delta(n)$-steal-resilient encryption implies that Φ_2 is $\delta(n)$-steal-resilient encryption, too.

6 Conclusion

In this work, we proposed a new and strong leakage setting, a novel notion of computational entropy, and a construction to achieve higher security against strong leakage. We separated our new notion from several relevant existing concepts, including Yao-Entropy, Hill-Entropy, All-or-Nothing Transform, Exposure Resilient Function. Unlike most of previous leakage resilient cryptography works which focused on defeating side-channel attacks, we opened a new direction to study how to defend against backdoor (or Trojan horse) and covert channel attacks.

Acknowledgment

The first author is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Corporate Laboratory@University Scheme, National University of Singapore, and Singapore Telecommunications Ltd. The second author is supported by the National Research Foundation (NRF), Prime Minister's Office, Singapore, under its National Cybersecurity R&D Programme (Award No. NRF2014NCR-NCR001-31) and administered by the National Cybersecurity R&D Directorate.

References

1. Abdalla, M., Belaïd, S., Fouque, P.A.: Leakage-resilient symmetric encryption via re-keying. In: Proceedings of the 15th International Conference on Cryptographic Hardware and Embedded Systems. pp. 471–488. CHES'13 (2013)
2. Alwen, J., Dodis, Y., Wichs, D.: Leakage-resilient public-key cryptography in the bounded-retrieval model. In: Proceedings of the 29th Annual International Cryptology Conference on Advances in Cryptology. pp. 36–54. CRYPTO '09, Springer-Verlag, Berlin, Heidelberg (2009), `http://dx.doi.org/10.1007/978-3-642-03356-8_3`
3. Alwen, J., Dodis, Y., Wichs, D.: Survey: Leakage Resilience and the Bounded Retrieval Model. In: Proceedings of the 4th International Conference on Information Theoretic Security. pp. 1–18. ICITS'09 (2010)
4. A.N.Kolmogorov: On tables of random numbers 207, 387–395 (November)
5. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K.: On the (Im)Possibility of Obfuscating Programs. J. ACM 59(2), 6:1–6:48 (2012)
6. Barak, B., Shaltiel, R., Wigderson, A.: Computational Analogues of Entropy. In: Approximation, Randomization, and Combinatorial Optimization.. Algorithms and Techniques. pp. 200–215 (2003)
7. Barwell, G., Martin, D.P., Oswald, E., Stam, M.: Authenticated encryption in the face of protocol and side channel leakage. Cryptology ePrint Archive, Report 2017/068 (2017), `https://eprint.iacr.org/2017/068`
8. Barwell G., Martin D.P., O.E.S.M.: Authenticated encryption in the face of protocol and side channel leakage. In: Advances in Cryptology – ASIACRYPT 2017. pp. 693–723 (2017)
9. Ben Othman, J., Mokdad, L.: Enhancing Data Security in Ad Hoc Networks Based on Multipath Routing. Journal of Parallel and Distributed Computing 70, 309–316 (2010)
10. Bronchain, O., Dassy, L., Faust, S., Standaert, F.X.: Implementing trojan-resilient hardware from (mostly) untrusted components designed by colluding manufacturers. In: Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security. pp. 1–10. ASHES '18, ACM, New York, NY, USA (2018), `http://doi.acm.org/10.1145/3266444.3266447`
11. Canetti, R., Dodis, Y., Halevi, S., Kushilevitz, E., Sahai, A.: Exposure-resilient Functions and All-or-nothing Transforms. In: Proceedings of the 19th International Conference on Theory and Application of Cryptographic Techniques. pp. 453–469. EUROCRYPT'00 (2000)
12. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard (2002)
13. Daniel Apon, Yan Huang, J.K.A.J.M.: Implementing Cryptographic Program Obfuscation. Cryptology ePrint Archive, Report 2014/779 (2014), `https://eprint.iacr.org/2014/779`
14. Di Crescenzo, G., Lipton, R., Walfish, S.: Perfectly Secure Password Protocols in the Bounded Retrieval Model. In: Proceedings of the Third Conference on Theory of Cryptography. pp. 225–244. TCC'06 (2006)
15. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Efficient Public-Key Cryptography in the Presence of Key Leakage. In: ASIACRYPT '10: ADVANCES IN CRYPTOLOGY. pp. 613–631 (2010)
16. Dziembowski, S.: Intrusion-Resilience via the Bounded-storage Model. In: Proceedings of the Third Conference on Theory of Cryptography. pp. 207–224. TCC'06 (2006)
17. Dziembowski, S., Faust, S., Standaert, F.X.: Private circuits iii: Hardware trojan-resilience via testing amplification. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 142–153. CCS '16, ACM, New York, NY, USA (2016), `http://doi.acm.org/10.1145/2976749.2978419`
18. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science. pp. 293–302. FOCS '08, IEEE Computer Society, Washington, DC, USA (2008), `http://dx.doi.org/10.1109/FOCS.2008.56`
19. Gabizon, A., Raz, R., Shaltiel, R.: Deterministic Extractors for Bit-Fixing Sources by Obtaining an Independent Seed. SIAM J. Comput. 36(4), 1072–1094 (2006)
20. Goldwasser, S., Kalai, Y.T.: On the impossibility of obfuscation with auxiliary input. In: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science. pp. 553–562. FOCS '05 (2005)
21. HÅsstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A Pseudorandom Generator from Any One-way Function. SIAM J. Comput. 28(4), 1364–1396 (1999)

22. Hsiao, C.Y., Lu, C.J., Reyzin, L.: Conditional Computational Entropy, or Toward Separating Pseudoentropy from Compressibility. In: Proceedings of the 26th Annual International Conference on Advances in Cryptology. pp. 169–186. EUROCRYPT '07 (2007)
23. Krawczyk, H.: Secret Sharing Made Short. In: Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology. pp. 136–146. CRYPTO '93 (1994)
24. McEliece, R.J., Sarwate, D.V.: On Sharing Secrets and Reed-Solomon Codes. Commun. ACM 24(9), 583–584 (Sep 1981)
25. Micali, S., Reyzin, L.: Physically observable cryptography (extended abstract). In: Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings. Lecture Notes in Computer Science, vol. 2951, pp. 278–296. Springer (2004)
26. Pereira, O., Standaert, F.X., Vivek, S.: Leakage-Resilient Authentication and Encryption from Symmetric Cryptographic Primitives. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. pp. 96–108. CCS '15 (2015)
27. Rabin, M.O.: Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance. J. ACM pp. 335–348 (1989)
28. Rabin, M.O.: Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance. Journal of the ACM 36(2), 335–348 (Apr 1989), `http://doi.acm.org/10.1145/62044.62050`
29. Rivest, R.L.: All-or-nothing encryption and the package transform. In: Proceedings of the 4th International Workshop on Fast Software Encryption. pp. 210–218. FSE '97 (1997)
30. Schwarz, M., Weiser, S., Gruss, D., Maurice, C., Mangard, S.: Malware guard extension: Using SGX to conceal cache attacks. CoRR abs/1702.08719 (2017), `http://arxiv.org/abs/1702.08719`
31. Shamir, A.: How to Share a Secret. Communications of the ACM 22(11), 612–613 (1979)
32. Standaert, F.X., Pereira, O., Yu, Y.: Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In: CRYPTO. pp. 335–352. Springer (2013)
33. Xu, J., Zhou, J.: Strong leakage resilient encryption by hiding partial ciphertext. Cryptology ePrint Archive, Report 2018/846 (2018), `https://eprint.iacr.org/2018/846`
34. Xu, J., Zhou, J.: Virtually isolated network: A hybrid network to achieve high level security. In: Data and Applications Security and Privacy XXXII. pp. 299–311. DBSec '18 (2018)
35. Yao, A.C.C.: Theory and applications of trapdoor functions. In: Proceedings of 23rd Annual Symposium on Foundations of Computer Science. pp. 80–91. EUROCRYPT '07 (1982)
36. Yu, Y., Standaert, F.X., Pereira, O., Yung, M.: Practical leakage-resilient pseudorandom generators. In: Proceedings of the 17th ACM Conference on Computer and Communications Security. pp. 141–151. CCS '10, ACM, New York, NY, USA (2010), `http://doi.acm.org/10.1145/1866307.1866324`

A Background

A.1 All-or-Nothing Transform: Rivest's Package Transform

To be self-contained, we *quote* the All-or-Nothing Transform, called "Package Transform", proposed by Rivest [29] as below.

1. Let the input message be $m_1, m_2, \ldots, m_s$.
2. Choose at random a key K' for the package transform block cipher $\mathsf{E}(\cdot,\cdot)$.
3. Compute the output message $m'_1, m'_2, \ldots, m'_{s+1}$ as below
   - $m'_i = m_i \oplus \mathsf{E}(K', i)$ for $i = 1, 2, \ldots, s$;
   - $m'_{s+1} = K' \oplus h_1 \oplus h_2 \ldots \oplus h_s$ where $h_i = \mathsf{E}(K_0, m'_i \oplus i)$ for $i = 1, 2, \ldots, s$.

Informally, in the above All-or-Nothing Transform method, if a receiver with key K_0 obtains all but a few ciphertext blocks m_i's, then the receiver will not be able to recover the random nonce key K' and thus can not decrypt any ciphertext block, i.e. know nothing about the plaintext.

B Our Proofs

B.1 Proof of Claim 1

Proof (Proof of Claim 1). For any non-negative integer ℓ, a steal algorithm $\mathsf{S}(\ell)$ can output any message in the set $\{0,1\}^{\leq \ell} \setminus \{\texttt{EmptyString}\}$, i.e. non-empty bit-string with length at most ℓ. The size of this set is

$$\sum_{i=1}^{\ell} 2^i = 2^{\ell+1} - 2. \tag{31}$$

Let $\mathcal{X}$ be a uniform random variable over $\{0,1\}^n$. Let a steal algorithm $\mathsf{S}(n-1)$ output $2^n - 2$ distinct messages, such that each message can encode a unique value of $\mathsf{P}(\mathcal{X})$, with two possible values (denoted as x_0 and x_1) of $\mathsf{P}(\mathcal{X})$ ignored. For any $x \in \{0,1\}^n \setminus \{x_0, x_1\}$, we have

$$\mathsf{Adv}^{\mathsf{out}}_{\mathcal{A}(n-1),\mathsf{P}}(x) = \Pr\left[\mathsf{R}\left(\mathsf{S}^{\mathcal{O}\left(y \leftarrow \mathsf{P}(x)\right)}(n-1)\right) = y\right] = 1. \tag{32}$$

Therefore,

$$\Pr_{x \xleftarrow{R} \{0,1\}^n}\left[\mathsf{Adv}^{\mathsf{out}}_{\mathcal{A}(n-1),\mathsf{P}}(x) = 1\right] = 1 - \frac{2}{2^n}. \tag{33}$$

Therefore, Claim 1 is proved by combining the above equation and the definition of steal-entropy in Eq (5).

B.2 Proof of Lemma 1

Proof (Proof of Lemma 1). Let c be any t-time compressor algorithm with output length $\ell \leq \xi$ and d be any t-time decompressor algorithm d. We construct a t-adversary $\mathcal{A}^*$ with steal algorithm S and recovery algorithm R such that: (1) S invokes the compressor algorithm c to compress the output of $\mathsf{P}(x)$ into ℓ bits message. From this ℓ bits message, R invokes the decompressor algorithm d to recover the value x. For adversary $\mathcal{A}^*$, for every ℓ, we define a subset $\mathbf{G}_\ell \subset \{0,1\}^n$ as

$$\mathbf{G}_\ell \overset{\text{def}}{=} \left\{x \in \{0,1\}^n : \mathsf{Adv}^{\mathsf{out}}_{\mathcal{A}^*(\ell),\mathsf{P}}(x) \leq \frac{1}{2^{\xi-\ell}} + \epsilon\right\} \tag{34}$$

From $\inf \mathbb{S}^{\mathsf{out}}_{\epsilon,t}(\mathsf{P}) \geq \xi$, we get $\Pr_{x \leftarrow \mathcal{X}}\left[x \in \mathbf{G}_\ell\right] \geq 1 - \epsilon$.

We can calculate the success probability of the compressor and decompressor as below:

$$
\begin{aligned}
&\Pr_{y\leftarrow \mathsf{P}(\mathcal{X})}\left[d(c(y))=y\right] \\
=\ &\Pr_{x\leftarrow \mathcal{X}}\left[d(c(\mathsf{P}(x)))=\mathsf{P}(x)\right] \\
=\ &\Pr_{x\leftarrow \mathcal{X}}\left[\mathsf{R}\left(\mathsf{S}^{\mathcal{O}\left(y\leftarrow \mathsf{P}(x)\right)}(\ell)\right)=y\right] \\
=\ &\Pr_{x\leftarrow \mathcal{X}}\left[\mathsf{Adv}^{\mathtt{out}}_{\mathcal{A}^*(\ell),\mathsf{P}}(x)\right] \\
=\ &\Pr_{x\leftarrow \mathcal{X}}\left[\mathsf{Adv}^{\mathtt{out}}_{\mathcal{A}^*(\ell),\mathsf{P}}(x)\ \middle|\ x\in \mathbf{G}_\ell\right]\times \Pr_{x\leftarrow \mathcal{X}}\left[x\in \mathbf{G}_\ell\right]+ \\
&\Pr_{x\leftarrow \mathcal{X}}\left[\mathsf{Adv}^{\mathtt{out}}_{\mathcal{A}^*(\ell),\mathsf{P}}(x)\ \middle|\ x\notin \mathbf{G}_\ell\right]\times \Pr_{x\leftarrow \mathcal{X}}\left[x\notin \mathbf{G}_\ell\right] && (35)\\
\le\ &\left(\frac{1}{2^{\xi-\ell}}+\epsilon\right)\times \Pr_{x\leftarrow \mathcal{X}}\left[x\in \mathbf{G}_\ell\right]+1\times \Pr_{x\leftarrow \mathcal{X}}\left[x\notin \mathbf{G}_\ell\right] && (36)\\
\le\ &\left(\frac{1}{2^{\xi-\ell}}+\epsilon\right)\times 1+1\times \Pr_{x\leftarrow \mathcal{X}}\left[x\notin \mathbf{G}_\ell\right] && (37)\\
\le\ &\left(\frac{1}{2^{\xi-\ell}}+\epsilon\right)+\epsilon && (38)
\end{aligned}
$$

B.3 Proof of Lemma 2

Proof (Sketch Proof of Lemma 2). Let algorithm P be a cryptographically secure pseudorandom number generator, with output length $m = poly(n)$. If a pair of efficient algorithms $c(\cdot), d(\cdot)$ can compress and uncompress the output of $\mathsf{P}(\mathcal{X})$, then these two algorithms $c(\cdot)$ and $d(\cdot)$ constitute an efficient distinguisher which can distinguish output of $\mathsf{P}(\mathcal{X})$ from true randomness, conflicting with assumption that P is cryptographically secure pseudo random number generator.

B.4 Proof of Lemma 3

Proof (Sketch Proof of Lemma 3). Let algorithm P be a cryptographically secure pseudorandom number generator, with output length $m = poly(n)$. This lemma can be easily proved by evaluating the steal-entropy of algorithm P and Hill-Entropy of variable $\mathsf{P}(\mathcal{X})$.

B.5 Proof of Lemma 5

Proof (Proof of Lemma 5). Let the algorithm P be an instance of exposure resilient function $g(x) = G(f(x))$ as in Lemma 4.6 in Canetti el al. [11], which is quoted as Lemma 4 in this paper, such that the parameters satisfy this condition: $n = \ell + poly(k)$. Clearly, the constructed function P is a computational ℓ-**ERF**. On the other hand, under our formulation, the attacker may simply steal k bits value $y = f(x)$ via backdoor and covert channel, and then compute and output all of m bits output $\mathsf{P}(x) = g(x) = G(y)$. Thus, the steal-entropy of in output $\sup \mathbb{S}^{\mathtt{out}}_{\epsilon,t}(g) \le k$.

B.6 Proof of Lemma 6

Proof (Proof of Lemma 6). **Construction of R_1.** We construct R_1 by repeatedly invoking R_0 in this way: Given input `Msg` and $y = \mathsf{P}(x)$, recovery algorithm R_1 makes N number of independent invocation on randomized algorithm $\mathsf{R}_0(\mathtt{Msg}, y)$ using independent random seeds, and obtains N outputs , denoted as $(\bar{x}^{(j)}, \mathbf{I}^{(j)})$, where $j \in [1, N]$. For each bit position $i \in [1, n]$, count how many sets $\mathbf{I}^{(j)}$, $j \in [1, N]$, contains element i and denote this count value as weight $w_i := |\{\mathbf{I}^{(j)} : i \in \mathbf{I}^{(j)}\}|$. Let $\mathbf{I}$ be the set of $(\ell + \varDelta)$ bit positions i's from $[1, n]$ with top $(\ell + \varDelta)$ largest weight w_i. For each $i \in \mathbf{I}$, make a majority vote on set $\{\bar{x}^{(j)}[i] : i \in \mathbf{I}^{(j)}\}$ of bit values, and denote the resulting bit as $\bar{x}[i]$. For each $i \notin \mathbf{I}$, randomly choose a bit and denoted it as $\bar{x}[i]$. R_1 will output $(\bar{x} = \bar{x}[1]..\bar{x}[n], \mathbf{I})$.

Claim 2 *Let $N = \Theta(1/\epsilon)$. If $\mathtt{Msg} \in \mathbf{G}^{\mathsf{R}_0}_{\mathtt{msg}}(\ell, \varDelta, x, 0.5+\epsilon)$, then $\mathtt{Msg} \in \mathbf{G}^{\mathsf{R}_1}_{\mathtt{msg}}(\ell, \varDelta, x, 1-negl(\lambda))$. In other words, $\mathbf{G}^{\mathsf{R}_0}_{\mathtt{msg}}(\ell, \varDelta, x, 0.5+\epsilon) = \mathbf{G}^{\mathsf{R}_1}_{\mathtt{msg}}(\ell, \varDelta, x, 1 - negl(\lambda))$*

Claim 2 could be proved easily using Hoeffding's Inequality and our definition of $\mathbf{G}_{\mathtt{msg}}$.

Construction of S_1. Make N' number of independent invocation of randomized algorithm $\mathsf{S}^{\mathcal{O}(\mathsf{P}(x))}$ and obtains output $\mathtt{Msg}_j$, $j \in [1, N']$. Loop from $j = 1$ upto N', invoke algorithm $\mathsf{R}_1(\mathtt{Msg}_j, \mathsf{P}(x))$ to obtain output $(\hat{x}^{(j)}, \mathbf{I}^{(j)})$. Check if the following two conditions hold: (1) the size of set $\mathbf{I}^{(j)}$ is at least $\ell + \varDelta$; (2) for each $i \in \mathbf{I}^{(j)}$, $\hat{x}^{(j)}[i] = x[i]$. If both of the above two conditions hold, then abort the loop and output $\mathtt{Msg}_j$. Otherwise, for any j, at least one of the above condition does not hold, then fail.

Claim 3 *Let $N' = \Theta(1/\epsilon)$. $\mathbf{G}^{\mathsf{S}_0,\mathsf{R}_0}_{\mathtt{x}}(\ell, \varDelta, 0.5+\epsilon, 0.5+\epsilon) = \mathbf{G}^{\mathsf{S}_0,\mathsf{R}_1}_{\mathtt{x}}(\ell, \varDelta, 0.5+\epsilon, 1 - negl(\lambda))$.*

Claim 3 can be easily proved using the result of Claim 2 and the definition of $\mathbf{G}_{\mathtt{x}}$: More precisely, just replace set $\mathbf{G}^{\mathsf{R}_0}_{\mathtt{msg}}(\ell, \varDelta, x, 0.5+\epsilon)$ with $\mathbf{G}^{\mathsf{R}_1}_{\mathtt{msg}}(\ell, \varDelta, x, 1 - negl(\lambda))$ in Equation 19.

Claim 4 *Let $N' = \Theta(1/\epsilon)$. $\mathbf{G}^{\mathsf{S}_0,\mathsf{R}_1}_{\mathtt{x}}(\ell, \varDelta, 0.5+\epsilon, 1 - negl(\lambda)) = \mathbf{G}^{\mathsf{S}_1,\mathsf{R}_1}_{\mathtt{x}}(\ell, \varDelta, 1 - negl(\lambda), 1 - negl(\lambda))$.*

Claim 2 could be proved easily using Hoeffding's Inequality and our definition of $\mathbf{G}_{\mathtt{msg}}$.

B.7 Proof of Theorem 7

Proof (Sketch Proof of Theorem 7). Let Enc be any semantic-secure bock cipher with block length equal to 128, and Cipher Block Chaining (CBC) mode is chosen to encryption multi-blocks long message. Let $\mathsf{P}(x)$ be the suffix of ciphertext $\mathsf{Enc}_k(\mathsf{LongMsg})$, by removing the first 128 bits from $\mathsf{Enc}_k(\mathsf{LongMsg})$. It is easy to prove the above theorem by analyzing the Steal-Entropy and Strong Steal-Entropy of algorithm P.

B.8 Proof of Lemma 8

Proof (Proof of Lemma 8). An adversary could obtain (e.g. via eavesdropping) *almost* all ciphertext blocks m_i's with $i \in \mathbf{S} \subset [1, s+1]$ and the size of set $\mathbf{S}$ is close to $s+1$, e.g. $|\mathbf{S}| = s - 10$ (assuming total bit-length of 11 ciphertext blocks is much larger than bit length of key K'). The adversary could choose to steal the short secret key K' via backdoor algorithm S and the covert channel, and decrypt all ciphertext block m_i's with $i \in \mathbf{S}$ using the key K', although with 11 ciphertext blocks missing. Therefore, by stealing a short key K', the adversary is about to obtain all most all message blocks m_i with $i \in \mathbf{S}$ except 10 or 11 missing message blocks. By definition of strong-steal entropy (respectively, rate) in input, the above adversary is a witness that Lemma 8 holds.

B.9 Proof of Corollary 10

Proof (Proof of Corollary 10). This Corollary can be proved by evaluating the infimum of strong steal-entropy of Φ_0 in input using Theorem 9. Note that $\inf \mathbb{S}^{\mathtt{sin}}_{\epsilon,t}(\textsc{Suffix}_{\Phi_0}) \geq \zeta$ trivially implies that Φ_0 is a $\delta(n)$-steal-resilient encryption, with $\delta(n) = \frac{1}{\rho\tau}$ by Definition 9 and the equality $n = \rho\zeta(1+\tau)$.

Next we will prove $\inf \mathbb{S}^{\mathtt{sin}}_{\epsilon,t}(\textsc{Suffix}_{\Phi_0}) \geq \zeta$ using proof by contradiction. Our hypothesis is that: $\inf \mathbb{S}^{\mathtt{sin}}_{\epsilon,t}(\textsc{Suffix}_{\Phi_0}) \geq \zeta$ does not hold. By Definition 7 (more precisely, Equation (22)), there exists a t-adversary $\mathcal{A} = (\mathsf{S}, \mathsf{R})$, such that

$$\forall \epsilon \geq \lambda^{-c}, \quad \Pr_{x \in_R \{0,1\}^n}[x \in \mathbf{G}^{\mathsf{S},\mathsf{R}}_{\mathtt{x}}(\ell, \varsigma(\ell,\epsilon) + 1 - \ell, 0.5+\epsilon, 0.5+\epsilon)] \geq 0.5 + 1/\mathit{poly}(\lambda) \tag{39}$$

According to Lemma 6, there exists another $t \cdot \Theta(1/\epsilon)$-adversary $\mathcal{A}' = (\mathsf{S}', \mathsf{R}')$ such that

$$\Pr_{x \in_R \{0,1\}^n}[x \in \mathbf{G}^{\mathsf{S}',\mathsf{R}'}_{\mathtt{x}}(\ell, \varsigma(\ell,\epsilon) + 1 - \ell, 1 - \mathit{negl}(\lambda), 1 - \mathit{negl}(\lambda))] \geq 0.5 + 1/\mathit{poly}(\lambda) \tag{40}$$

For any $x \in \mathbf{G}^{\mathsf{S}',\mathsf{R}'}_{\mathtt{x}}(\ell, \Delta, \alpha, \beta)$, let $(\bar{x}, \mathbf{I})$ denotes the output of recovery algorithm $\mathsf{R}(\mathtt{Msg}, \mathsf{P}(x))$, we have

$$\begin{aligned}\forall i \in \mathbf{I}, \Pr[\bar{x}[i] = x[i]] &= \Pr\left[\bar{x}[i] = x[i] \mid \mathtt{Msg} \in \mathbf{G}^{\mathsf{R}}_{\mathtt{msg}}(\ell, \Delta, x, \beta)\right] \times \Pr\left[\mathtt{Msg} \in \mathbf{G}^{\mathsf{R}}_{\mathtt{msg}}(\ell, \Delta, x, \beta)\right] \\ &\geq \alpha \cdot \beta = (1 - \mathit{negl}(\lambda)) \times (1 - \mathit{negl}(\lambda)) \geq 1 - 2 \times \mathit{negl}(\lambda). \end{aligned} \tag{41}$$

This means that, the polynomial time adversary $\mathcal{A}' = (\mathsf{S}', \mathsf{R}')$ could steals at most $\ell \leq \zeta - 2$ bits of message and output $\ell + 2 \leq \zeta$ bits of information of x (i.e. $x[i]$ for $i \in \mathbf{I}$) with overwhelming high probability $1 - 2 \times \mathit{negl}(\lambda)$, with at least $0.5 + 1/\mathit{poly}(\lambda)$ fraction of input x in the domain $\{0,1\}^n$.

According to Theorem 9, and our argument after Property 1 on page 10, any ζ distinct bits $x[j]$ together will have ζ bits joint-Shannon entropy.

So any $\ell + 2$ bits of $x[i]$'s for at least $0.5 + 1/\mathit{poly}(\lambda)$ fraction of input x in the domain $\{0,1\}^n$, will have joint-Shannon entropy at least

So collection of $x[i]$'s in the output of R' will have joint Shannon-entropy at least

$$\log\left(2^{\ell+2} \times (1 - 2 \times \mathit{negl}(\lambda)) \times \left(0.5 + 1/\mathit{poly}(\lambda)\right)\right) \geq \ell + 1. \tag{42}$$

However $\mathsf{S}'(\ell)$ could only encode $\sum_{i=1}^{\ell} 2^i = 2^{\ell+1} - 2$ distinct messages, it is a contradiction and the hypothesis does not hold.