On Hardening Leakage Resilience of Random Extractors for Instantiations of Leakage Resilient Cryptographic Primitives

Danyang Chen[a,b], Yongbin Zhou[a,*], Yang Han[a], Rui Xue[a], Qing He[b]

[a] *State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences Beijing, China, 100195*
[b] *School of Mathematical Sciences, Beijing Normal University, Beijing, China, 100875*

Abstract

Random extractors are proven to be important building blocks in constructing leakage resilient cryptographic primitives. Nevertheless, recent efforts showed that they are likely more leaky than other elementary components (e.g. block ciphers) in unprotected implementations of these primitives, in the context of side-channel attacks. In this context, from the adversary's point of view, the extractors themselves could become the point of interest. This paper extends the problem of how leakage resilience of random extractors could be to the case of protected instantiations. Specifically, we investigate the feasibility of applying classical countermeasures to ameliorate leakage resilience of cryptographic components and/or primitives against side-channel attacks, and then show how to evaluate the physical leakage resilience of these instantiations theoretically and practically. The countermeasures we consider are masking, shuffling, and combination of them. Taking one leakage-resilient stream cipher presented at FOCS 2008 as a case of study, we not only examine the leakage resilience of the underlying extractor, but also discuss how leakages from the extractor and from the underlying pseudo-random generator respectively impact the leakage resilience of

*Corresponding Author

Email addresses: chendanyang@is.iscas.ac.cn (Danyang Chen), zhouyongbin@iie.ac.cn (Yongbin Zhou), hanyang@is.iscas.ac.cn (Yang Han), xuerui@iie.ac.cn (Rui Xue), heqing@bnu.edu.cn (Qing He)

the stream cipher as a whole. On the one hand, our theoretical and experimental results, which are consistent with each other, do justify some existing observations. On the other hand, and more importantly, our results reveal some new observations that contrast with these knowing ones, which explicitly indicates that previous observations are (mostly likely) incomplete. We argue that our work is of both obvious theoretical interest and important practical significance, and may help foster the further research on the design and implementation of random extractors in leakage-resilient cryptography.

Keywords:
Side-Channel Attacks, Random Extractor, Cryptographic Instantiations, Leakage Resilience, Masking and Shuffling

---

1. Introduction

Due to the prevalence of side-channel attacks in which the physical characteristics of a computation can be physically observed and measured, one has to carefully consider the physical security of implementations of cryptographic primitives even provably secure under traditional black-box model, especially when they are being instantiated on computing devices. Since some of these leakages (e.g. execution time, power consumption, electromagnetic emanation, acoustics, etc.[7],[5],[6]) are inevitably present in almost any physical implementation, it is widely believed that the problem of achieving cryptographic security on devices that leaks information about the internal secret states of cryptographic schemes cannot just be addressed by physical countermeasures alone. In order to deal with this problem, cryptology community began to consider the possibility of taking such leakages into consideration during the design of the mathematical specification of cryptographic primitives and of providing some formal and provable security thereof. Leakage resilient cryptography is one of such particularly meaningful attempts along this direction.

Basically, leakage resilient cryptography (LRC for short) aims to provide provable security against a wide spectrum of side-channel attacks. The original idea of LRC dates back to the year of 2004[20]. After that, a number of theoretical results about the design and analysis of leakage resilient primitives have been presented, including symmetric schemes [4] [14], asymmetric schemes [3] [1] [11], and pseudo-random generators [21]. These works represent some important academic steps towards building sound theoretical

foundations of side-channel security.

On the other hand, however, for these theoretically leakage resilient constructions to be meaningful in practice, they have to be instantiated and implemented on some specific hardware platforms. Taking this important point into consideration, it is pivotal to investigate and to evaluate the practical leakage resilience of various instantiations, so that the physical security of different implementations could be objectively compared or that a more secure implementation could be identified among others. Actually, it has already been turned out that there really exist some gaps between theoretically leakage resilient constructions and specific hardware implementations of them [14]. As a natural consequence, the problem of how to analyze the practical leakage resilience of a cryptographic instantiation and then to harden it in some specific implementations becomes very relevant and interesting.

Among other existing building blocks, random extractors are proven to be extremely important tools in constructing leakage resilient cryptographic primitives. For example, they have been successfully employed in fabricating both symmetric and asymmetric primitives [8]. Unfortunately, recent efforts showed that random extractors are more leaky than other components such as block ciphers in unprotected implementations of these primitives. In this context, from the adversary's point of view, the extractors themselves could likely become the point of interest, or the point of attack. Additionally, it has been recognized to be difficult (if not impossible) to discuss the physical security of a cryptographic primitive without fully understanding the relations between its algorithmic description and its implementation properties. Among many other relevant issues, the applicability of classical countermeasures, such as masking and hiding, to extractor implementations remains to be a problem of both theoretical interest and practical significance.

Motivated by these, this paper extends this problem to the case of protected instantiations.Specifically, we investigate the feasibility of applying classical countermeasures to ameliorate the leakage resilience of cryptographic components and/or primitives against side-channel attacks, and then show how to evaluate the physical leakage resilience theoretically and practically. The countermeasures we consider are masking, shuffling, and combination of them. Taking one leakage-resilient stream cipher presented at FOCS 2008 (LR-SC for short) as a case of study, we not only examine the leakage resilience of the underlying extractor (Ext for short), but also discuss how leakages from the extractor and from the underlying pseudo-random generator (PRG for short) respectively impact the leakage resilience of the leakage-

resilient stream cipher as a whole.

Theoretically, we use min-entropy[①] as a quantitative metric to capture the physical leakage resilience of implementations of the extractor. To harden the leakage resilience of these implementations, masking, shuffling, and the combination of them are considered. All software implementations we considered are running on a typical 8-bit microcontroller STC89C58RD+. On the one hand, our theoretical and experimental results are consistent with each other, and justify some existing observations. On the other hand, and more importantly, our results reveal some new observations that contrast with knowing ones, which explicitly indicates that previous observations are (mostly likely) incomplete. Our results may help foster the further research on the design and implementation of random extractors in leakage-resilient cryptographic constructions, even though our analysis mainly focuses on various instantiations of the random extractor used in leakage resilient stream cipher of [4].

1.1. Related Work

Since the work of leakage resilient cryptography at FOCS 2008 [4], a number of positive results on the constructions of leakage resilient primitives have appeared [4] [14] [3] [1] [11]. For example, based on the "Only Computation Leaks" (OCL for short) assumption, [4] constructed a leakage resilient stream cipher using alternating extraction. The main idea of [4] is to split the secret information into two independent parts. In each round of the secret-dependent computation, one and only one part of the two secret parts is accessed, while the other part remains unaccessed. According to the OCL assumption, half of the secret that is not accessed will leak no information. In this way, [4] proved that the output sequence of LR-SC is close to uniform randomness (see Lemma1 in Section 2), and thus its provable security.

Afterwards, [14] considered the practical security of the scheme of [4], and instantiated the LR-SC using the following two-source extractor and length-tripling PRG based on block cipher.

A length-tripling PRG based on a block cipher, say AES for instance, is constructed as follows,

$$PRG : \{0,1\}^n \mapsto \{0,1\}^{3n} : x \mapsto (AES_x(c_1), AES_x(c_2), AES_x(c_3))$$

[①] We argue that min-entropy is more relevant than typical Shannon entropy in our case, because the original definition of random extractor is based on the former.

where c_1, c_2 and c_3 are constants.

A two-source extractor which allows extracting many random bits from two weak sources, is defined to be

$$Ext : \{0,1\}^l \times \{0,1\}^l \mapsto \{0,1\}^n : (x,y) \mapsto ((A_1x)\cdot y, (A_2x)\cdot y, \cdots, (A_nx)\cdot y)$$

where $A_1, A_2, \ldots, A_n$ are $l \times l$ cyclic shift matrices over $GF[2]$, $\cdot$ is the inner product mod 2, and A_ix is a matrix-vector multiplication over $GF[2]$. In this extractor, l needs to be a prime having 2 as primitive root. Specifically we set $n = 128$, $l = 193$, but we only used the first 192 bits of x and y.

[14] implemented the extractor and the PRG on an 8-bit device and used DPA to attack them respectively. Equivalently, attacking the PRG equals attacking AES three times with different plaintexts. Figure 1 in Section 3 presents the computation of the extractor in one masked implementation. The illustrated process is to repeat 128 times in order to produce a PRG input (i.e. an AES key). In [14], the authors used the HW model and attacked the intermediate z_i^j, i.e. they assume that adversary can get $l_i^j = HW(x_i^j \odot y^j)$ from the sampled power traces, where j is the index for the 24 bytes of z_i and i is the index of A_i. The result of [14] shows that the extractor is the weak point of the LR-SC scheme, because its extensive computation leads to larger leakage. Therefore, [14] argued that it is unlikely that any extractor will be able to provide high security levels (especially in small devices) without being combined with other countermeasures.

Note that another independent work in [10], similar to ours, was done almost at the same time as the writing time of the main part of this work[②]. On the one hand, these two independent works share something in common. For example, both of them consider simulated power traces and target the same extractor. On the other hand, however, ***these two independent works actually differ a lot in many aspects from each other***. Their implementations are in (192-bit) hardware (where the specification is not provided), whereas ours are in software on 8-bit microcontroller STC89C58RD+.

[②]The main part of this work was completed in May 2011, and was then submitted to **ICICS 2011**(Thirteenth International Conference on Information and Communications Security), with the **Submission ID** being **NN07E34G201**. Regretfully, our submission was not accepted. The work in [10] first came to public at IACR ePrint Archive on June 27, 2011 with a Report ID 2011/348. Therefore, our work in this paper and that in [10] were done independently almost at the same time.

They considered parallelism in their implementations, whereas we consider sequentialism alone. They considered masking alone, whereas we considered much more than that, i.e. masking, shuffling, and combination of them. They performed higher-order DPA attacks in their experiments to challenge the protected implementations, whereas we adopted template-based DPA attacks, which are widely accepted to be most powerful and outperform the former [13]. Mutual information is used in their theoretical evaluation, while min-entropy in our work. The adoption of min-entropy implies our results are (likely) more pertinent to the security notion of random extractor, and thus are more reasonable and more rigorous. In addition, the evaluation of mutual information is solely for the verification of theoretical analysis results in our work. They considered 4-bit (only 1/48 of the word size of their hardware) key guess, whereas we considered 8-bit (which is exactly the word size of microcontroller STC89C58RD+) key guess. They considered *1, 2, 3-order* masking in masked implementations, while we only considered *1-order* masking (Note that we use template-based DPA attacks instead). They considered uniform distributed and biased distributed masks, whereas we only consider the former. They treated the number of leakage samples per plaintext exploited in the attacks to be a parameter, while we consider all leakage samples per plaintext (which is the worst case). Putting together all these above-mentioned aspects, we firmly believe that these two independent works are constructively complementary to each other.

1.2. Our Contributions

In this paper, we study the problem of how the theoretical and practical leakage resilience of random extractors would be in the scenarios of protected instantiations, and then to what extend and how it would be hardened by applying traditional countermeasures. Specifically, we investigate the feasibility of applying classical countermeasures, such as masking, shuffling, and combinations of them, to improve the leakage resilience of cryptographic components and/or primitives against side-channel attacks. Our main contributions are three-fold as follows.

- ***Theoretically***, we demonstrate how to use pertinent information-theoretic metric, such as min-entropy, to capture the theoretical leakage-resilience of instantiations of typical extractors. For this purpose, min-entropy is used to capture the pseudo-randomness of the outputs of various instantiations of extractors. Taking one leakage-resilient stream

cipher presented at FOCS 2008 as a case of study, we not only examine the leakage resilience of the underlying extractor, but also discuss how leakages from the extractor and from the underlying pseudo-random generator respectively impact the leakage resilience of the stream cipher as a whole.

- **Practically**, we perform six sets of well-designed template based DPA attacks on multiple extractor implementations and show how to examine the practical leakage-resilience of these instantiations. For this purpose, mutual information for four extractor implementations is evaluated to verify the theoretical analysis results. In our protected implementations, masking, shuffling, and combinations of them are considered.

- ***Instructively***, our theoretical and practical results reveal some new observations that are not embraced in existing ones, which explicitly means that previous observations are (mostly likely) incomplete. For one example, it is claimed, in [14], that "implementation of the extractor can become a better target for a DPA than an AES-based PRG" (which means that extractor could be a weaker point of the LR-SC as a whole), and in [10] this claim is re-clarified. Our experiment results show that for instantiations of extractor with masking (resp. shuffling) alone, this conclusion holds. However, this is NOT TRUE for instantiations of the extractor with both masking and shuffling. In the latter, it is harder for the adversary to attack the extractor than to attack the PRG. In other words, our new observation evidently shows that this claim is not always true. This important observation means that the analysis of practical security of random extractor may be more perplex than ever imagined, and may need more clarifications if possible.

2. Preliminaries

We will briefly introduce some elementary notations, definitions and lemmas used throughout the paper.

2.1. Notations and Definitions

Random variable X and Y take values in a finite set A, and their *statistical distance* is $d(X,Y) = \frac{1}{2}\sum_{a\in A}|Pr[X=a] - Pr[Y=a]|$. If X is distributed over $\{0,1\}^n$, then let $d(X) = d(X,U_n)$ denote the statistical distance of X

from a uniform distribution U_n over $\{0,1\}^n$. The *min-entropy* of a random variable X is $H_\infty(X) = -\log(\max_x Pr[X = x])$. It measures the worst-case predictability of X. The *average conditional min-entropy* of X given Z is defined by $\tilde{H}_\infty(X|Z) = -\log(\mathbb{E}_{z\leftarrow Z}[2^{H_\infty(X|Z=z)}])$, and it measures the worst-case predictability of X by an adversary that may observe a correlated variable.

Let $b \geq 0$, $\epsilon \geq 0$. A (b,ϵ)-*extractor* is an efficient function EXT: $\{0,1\}^{l_1} \times \{0,1\}^{l_2} \mapsto \{0,1\}^n$, such that for all random variable X over $\{0,1\}^{l_1}$ with min-entropy $H_\infty(X) \geq b$, and the uniform distribution U_{l_2} over $\{0,1\}^{l_2}$, we have $d(EXT(X,U_{l_2})) \leq \epsilon$.A (b_X,b_Y,ϵ)-*two-source extractor* is an efficient function TWO-EXT: $\{0,1\}^l \times \{0,1\}^l \mapsto \{0,1\}^n$, such that for all random variable X and Y over $\{0,1\}^l$ with min-entropy $H_\infty(X) \geq b_X$, $H_\infty(Y) \geq b_Y$, we have $d(TWO-EXT(X,Y)) \leq \epsilon$. A (b_X,b_Y,ϵ)-*strong blender* is an efficient function BLE: $\{0,1\}^l \times \{0,1\}^l \mapsto \{0,1\}^n$, such that for all random variable X and Y over $\{0,1\}^l$ with min-entropy $H_\infty(X) \geq b_X$, $H_\infty(Y) \geq b_Y$, we have $d(BLE(X,Y)|Y) \leq \epsilon$, i.e. given Y, statistical distance of $BLE(X,Y)$ from uniform distribution over $\{0,1\}^n$ is less than or equal to ϵ.

From the definitions above we can see that if a function is a two source extractor, then it is a extractor. If a function is a strong blender, then it is a two source extractor. That is, $\{StrongBlender\} \subseteq \{TwoSourceExtractor\} \subseteq \{Extractor\}$.

2.2. Lemmas and Corollary

Lemma 1 (Alternating Extraction)[4]*: Let EXT:* $\{0,1\}^l \times \{0,1\}^l \mapsto \{0,1\}^n$ *be a* (b,ϵ_{ext})*-extractor. Let* $A,B,K_0 \in \{0,1\}^l$ *be random variables where* A *and* B *are independent from each other and*

$$d(K_0|B) \leq \epsilon_0, H_\infty(A) \geq l-\Delta, H_\infty(B) \geq l-\Delta$$

Consider any $\lambda,\Delta,l \geq 0$ *and* $0 \leq \epsilon_{gap} \leq 1$ *which satisfy*

$$b \leq l - \Delta - \lceil r/2 \rceil(\lambda+n) - \log(1/\epsilon_{gap})$$

where r *is the round number,* λ *is the size of leakage,* Δ *is the lost of min-entropy of* A *and* B*. In the following inequality,* $\tau_r = B$ *if* r *is odd, and* $\tau_r = A$ *otherwise. We have*

$$d(K_{r+1}|K_0,\ldots,K_r,leakage_1,\ldots,leakage_r,\tau_r) \leq (r+1)\epsilon_{ext} + 2\epsilon_{gap} + \epsilon_0$$

i.e. given τ_r, all the outputs and all the leakage after the computation of K_r, the next key $K_{r+1} = EXT(K_r, \tau_r)$ to be output by S is $r\epsilon_{ext} + 2\epsilon_{gap} + \epsilon_0$-close to uniformly random.

Lemma 2 (Metric/HILL Pseudo-entropy of a PRG) [4]: *Let prg: $\{0,1\}^n \mapsto \{0,1\}^m$ and $f : \{0,1\}^n \mapsto \{0,1\}^\lambda$ (where $1 \leq \lambda < n < m$) be any efficiently computable functions. If prg is a $(\epsilon_{prg}, s_{prg})$-secure pseudorandom generator, then for any $\epsilon, \Delta > 0$ such that $\epsilon_{prg} \leq \frac{\epsilon^2}{2^\lambda} - 2^{-\Delta}$, with $X \sim U_n$ we have*

$$Pr_{y=f(x)}[H^{Metric}_{\epsilon, s_{prg}}(prg(X)|f(X) = y) \geq m - \Delta] \geq 1 - \epsilon$$

And for any $\epsilon_{HILL} > 0$

$$Pr_{y=f(x)}[H^{HILL}_{\epsilon+\epsilon_{HILL}, \hat{s}}(prg(X)|f(X) = y) \geq m - \Delta] \geq 1 - \epsilon$$

where $\hat{s} \approx s^2_{HILL} s_{prg}/8m$.

Lemma 3 (Strong Blender) [2]: *Let $A_1, A_2, \ldots, A_n$ be $l \times l$ matrices over $GF[2]$, such that for every non-empty subset $S \subset [k]$, the rank of $A_s = \sum_{l \in S} A_l$ is at least $l - t$, for some $0 \leq t < l$.*

$$\begin{array}{rccl} BLE_A: & \{0,1\}^l \times \{0,1\}^l & \mapsto & \{0,1\}^n \\ & (x, y) & \mapsto & ((A_1 x) \cdot y, \ldots, (A_n x) \cdot y) \end{array}$$

where $\cdot$ is the inner product mod 2, and $A_i x$ is a matrix-vector multiplication over $GF[2]$. Then the function BLE_A is a (b_X, b_Y, ϵ)-strong blender, with $\log(\frac{1}{\epsilon}) = \frac{b_X + b_Y + 2 - (l+t+n)}{2}$, that is

$$d(BLE_A(X, Y)|Y) = d((Y, BLE_A(X, Y)), (Y, U_n)) \leq 2^{-\frac{b_X + b_Y + 2 - (l+t+n)}{2}}$$

In [16], it has been proved that if $A_1, A_2, \ldots, A_n$ are right cyclic shift matrices, and l is a prime with 2 as a primitive root, then the rank of $\sum_{i=1}^n A_i$ is larger than or equal to $l - 1$, i.e. $t = 1$ in the inequality above.

According to the definitions of extractor and strong blender, the strong blender presented in Lemma 3 is essentially an extractor. The LR-SC in [4] makes use of an extractor and a pseudorandom generator. In the rest part of this paper, we use the extractor given in Lemma 3 and an AES-based pseudorandom generator to instantiate the LR-SC. Note that [14], [10] also

used these two constructs. Our theoretical analysis is based on the definition of the extractor in Section 2.1 and on that of the strong blender presented in Lemma 3. To analysis the security of the pseudorandom generator(i.e. AES), we use the following corollary.

Corollary 1 (Security of AES) *By combining Lemma 2, 4, 11 in [12], one can draw the conclusion that for AES with s rounds, for any PPT adversary $\mathscr{A}$, the advantage to distinguish the output of AES from any uniform randomness is*

$$Adv(\sum(F_1, \ldots, F_s)) \leq (kd^2 \cdot 2^{-m/k})^{\lfloor s/3 \rfloor}$$

where F_i is the transformation of the i^{th} round AES, d is the number of plaintexts, k is the number of bytes of key, and m is the number of bits of plaintext.

3. Security Analysis of Multiple Implementations of the Extractor

In this section, we will investigate the leakage resilience (aka physical security in this paper) of multiple implementations of the Extractor. Using multiple instantiations of the extractor with different countermeasures, we will show how to analyze to what extend the extractor has been hardened.

For this purpose, we use min-entropy to analyze the pseudo-randomness of the output of the extractor.For a (b, ϵ)-extractor Ext, if ϵ depends on b, i.e. there exists a function f, s.t. $\epsilon = f(b)$, then the value of b can be used to determine the security of the extractor. If $\epsilon = f(b) < 1$, then the extractor can be considered secure. If $\epsilon = f(b) \geq 1$, then it is hard to say whether $d(Ext(x, y))$ is less than 1 or not. In the latter case, we cannot say that the extractor is secure. Let $S = \{b | \epsilon = f(b) < 1\}$. If $b \in S$, then the extractor is secure; otherwise, it is insecure.

Next, taking the extractor used in [14] as a case study, we will concretely investigate the security of multiple implementation of the extractor, and they are unprotected implementation, masked implementation, shuffled implementation, masked and shuffled implementation respectively. For the purpose of simple yet clear illustration, we will take into consideration power analysis attacks alone, even though other side-channel attacks may also matter. In addition, it is assumed that the Hamming Weight of any interesting intermediate is available to the adversary, as presented in Figure 1. To simplify the analysis, we assume the adversary only gets the leakage of the first

elementary operation[③], i.e. she only gets $HW(z_i^j), where j \in [1:24]$. We will show that even this small leakage could lead to serious threats to the security of the extractor. The results are summarized in Table 1.

3.1. Unprotected Implementation

The unprotected implementation of the extractor is a naive implementation without any countermeasures. In this case, according to Lemma 3, ϵ is dependent on $b : \epsilon = 2^{-\frac{b_X+b_Y+2-(l+r+n)}{2}}$.

Without leakage, $b_x = b_y = 192$. According to Lemma 3, $d(Ext(x,y)|x) \leq 2^{-\frac{b_X+b_Y+2-(l+r+n)}{2}} = 2^{-32.5}$. The output of the extractor is quite close to uniform randomness. After leakage, the min-entropy of x and y decrease to $b_x = b_y = 111.3361$[④]. According to Lemma 3, $d(Ext(x,y)|x) \leq 2^{48.1639}$, which is quite a large distance from uniform randomness. Therefore, the unprotected implementation is not secure in terms of min-entropy. If we want $\epsilon \leq 1$, then we need $b_x + b_y \geq 319$, so $S = \{(b_x, b_y)|b_x + b_y \geq 319\}$.

3.2. Masked Implementation

Masking is a kind of algorithmic countermeasure, and it essentially belongs to secret sharing scheme. In concrete implementation, one usually uses one or more random value(s) to XOR one sensitive intermediate value in order to randomize the intermediate value. The number of random values used is called the order of masking. If one single random values is used, then the schema is called first order masking. If two or more random values are used, it is called higher-order masking.Let d denote the order of masking. In this paper, we consider $d = 1$ only.

Our masked implementation of the extractor is presented in Figure 1, where $m_i^j \leftarrow \{0,1\}^8$, $i \in [1:128]$, $j \in [1:24]$. When computing z_i^j, first compute $z'^j_i = x_i^j \odot (m_i^j \oplus y^j) = (x_i^j \odot m_i^j) \oplus (x_i^j \odot y^j) = m'^j_i \oplus z_i^j$. After the whole computation of b_i, we compute $b_i = b'_i \oplus \left(\oplus_{j=1}^{24} \oplus_{k=1}^{8} m'^j_i(k)\right)$ to remove the mask, where $m_i^j(k)$ denotes the k^{th} bit of m_i^j.

Considering the fact that template based DPA attacks [9] are widely accepted to be the most powerful [13], we perform template based DPA

[③]For random extractor, an elementary operation is the computation of one bit of its output in this paper.

[④]b_x (resp. b_y) is the average conditional min-entropy of x (resp. y) given $HW(z_i^j), j \in [1:24]$.

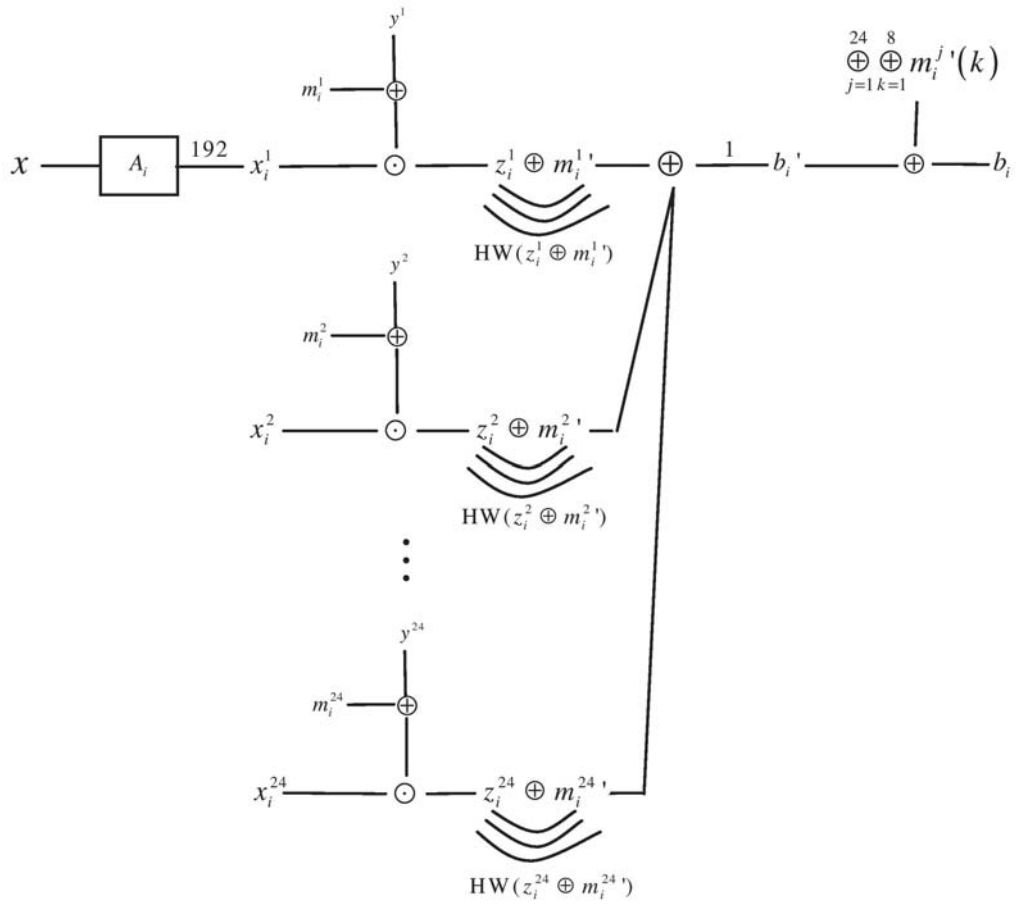

Figure 1: Masked Implementation of the Extractor

attacks against our masked implementation of the extractor. In this case, one builds template for pair of $HW(m)$ and $HW(x \odot (y \oplus m))$. After these two leakages, the min-entropy of x and y decrease to $b_x = 111.3361$, $b_y = 167.3544$. Consequently, $b_x + b_y = 278.7004 < 319$, which directly means that the masked implementation is not secure in terms of min-entropy either.

3.3. Shuffled Implementation

Shuffling is a kind of hiding technique. The basic idea of this approach is to randomly arrange the execution sequences of some independent operations. The number of independent operations to be randomized is called shuffling degree, denoted by N. For example, in the computation of b_1 in the extractor,

for all 24 bytes, the computation of $\oplus$ (AND) is independent from each other. Therefore, we could naturally consider adopting shuffling to decrease the adversary's ability of exploiting the leakages from the power traces.

Surprisingly, it turns out that shuffling operations does ***NOT*** change the min-entropy of x or that of y at all. For example, in Table 1, the min-entropy of shuffled implementation equals that of unprotected implementation, and the min-entropy of masked implementation is identical with that of masked and shuffled implementation. This is due to the fact that min-entropy just measures the worst-case predictability. For our present context, the worst-case of shuffling is that the output after shuffling remains the same as that without shuffling. So the min-entropy of x and that of y remain unchanged, which consequently means that ϵ remains unchanged. Note that it is really harder to attack shuffled implementations in real world than to attack unprotected implementations. This is because the average-case predictability does get lower after shuffling, although the worst-case predictability remains unchanged. Obviously, the shuffled implementation is insecure in terms of min-entropy.

3.4. Masked and Shuffled Implementation

In this subsection, we will apply both masking and shuffling into the implementation of the extractor. Recalling that shuffling does NOT help decrease the min-entropy, the min-entropy of y of masked and shuffled implementation will be identical with that of masked implementation. Therefore, masked and shuffled implementation is not secure in terms of min-entropy, either.

Table 1: Min-Entropy and $\log(\frac{1}{\epsilon})$ of Implementations of the Extractor

Implementations	Min-Entropy			$\log(\frac{1}{\epsilon})$
	b_x	b_y	$b_x + b_y$	
Unprotected	111.3361	111.3361	222.6722	48.1639
Shuffled, $N = 24^{\ddagger}$	111.3361	111.3361	222.6722	48.1639
Masked, $d = 1$	111.3361	167.3544	278.7004	20.1498
Masked and Shuffled, $d = 1, N = 24^{\ddagger}$	111.3361	167.3544	278.7004	20.1498

‡ Note that Min-Entropy is regardless of N.

4. Security of the LR-SC as a Whole

In this section, we will discuss how leakages from the extractor and from PRG will influence the entire security of the whole LR-SC scheme in [DP2008] as a whole.

4.1. Security of the Extractor

Lemma 1 in Section 2 requires that $H_\infty(y) \geq l - \Delta$, $H_\infty(x) \geq l - \Delta$. Since $b_x = b_y = 111.3460$, $l - \Delta \leq 111.3460$. The amount of leakages is $\lambda = 62.5320$[5]. $0 \leq \epsilon_{gap} \leq 1$, so $\log(1/\epsilon_{gap}) \geq 0$. Furthermore, Lemma 1 also requires that

$$\begin{array}{rcl} b_x & \leq & l - \Delta - \lceil r/2 \rceil(\lambda + n) - \log(1/\epsilon_{gap}) \\ & \leq & 111.3460 - (62.5320 + 128) - \log(1/\epsilon_{gap}) \\ & = & -79.1860 - \log(1/\epsilon_{gap}) \end{array}$$

Since $b_x = 111.3460$, the above requirement cannot be satisfied. Therefore, the pseudo-randomness of the output of the LR-SC as a whole is not theoretically guaranteed.

In our shuffled implementation, $l - \Delta \leq \min\{b_x, b_y\} = 180.6868$, the requirement becomes $b_x \leq -9.8452 - \log(1/\epsilon_{gap})$; in our masked implementation, $l - \Delta \leq \min\{b_x, b_y\} = 111.3460$, the requirement becomes $b_x \leq -79.1860 - \log(1/\epsilon_{gap})$; in our masked and shuffled implementation, $l - \Delta \leq \min\{b_x, b_y\} = 111.3460$, the requirement becomes $b_x \leq -79.1860 - \log(1/\epsilon_{gap})$. Whatever, all above four requirements cannot be satisfied. Therefore, for all these four implementations, due to leakages from the extractor, the LR-SC as a whole is NOT theoretically secure.

4.2. Security of the PRG

We consider the advantage of any PPT adversary in distinguishing the output of AES from uniform randomness. According to Corollary 1, $m = 128$, $k = 16$, $s = 10$, $d = 3$, then we have

$$Adv(\sum(F_1, \ldots, F_s)) \leq (kd^2 2^{-m/k})^{\lfloor s/3 \rfloor} = (16 \times 3^2 \times 2^{-128/16})^{\lfloor 10/3 \rfloor} = 3^6 \cdot 2^{-12}$$

We only consider unprotected implementation of PRG in this subsection. Lemma 2 in Section 2 requires that $\epsilon_{prg} \leq \frac{\epsilon^2}{2^\lambda} - 2^{-\Delta}$. According to the

[5] 62.5320 equals the product of average code length of HW and 24 (bytes).

computation above, we have $\epsilon_{prg} = 3^6 \cdot 2^{-12}$. To attack AES, HWs of 16 intermediates are needed. Then the leakage $\lambda = (1 - (1 - 41.6880)^3) = 88.7432$[6]. Therefore, $3^6 \cdot 2^{-12} \leq \frac{\epsilon^2}{2^{88.7432}} - 2^{-\Delta}$. However, such ϵ, Δ do not exist. Consequently, the AES-based PRG cannot meet the requirement of Lemma 2, which means its output does NOT preserve high min-entropy after some of its inputs are leaking. With respect to this fact, the PRG will not be a theoretically secure component for the LR-SC as whole, even though the PRG itself may not be the weakest point.

5. Experiments

In this section, we will present our theoretical and practical security evaluation and experimental results, in order to verify the relevance of those results presented in Section 3. In theoretical side, we will study the leakage resilience of Ext(resp. PRG) by computing the mutual information of its respective implementations. In practical side, we will investigate the leakage resilience of Ext(resp. PRG) by performing template based DPA attacks against multiple implementations of Ext(resp. PRG) respectively. Moreover, our experiments are based on simulated power traces.

Specifically, we examine multiple instantiations of Ext(resp. PRG), and they are unprotected implementation, masked implementation, shuffled implementation, and masked and shuffled implementation respectively. All of these instantiations of Ext(resp. PRG) are software implementations on an 8-bit microcontroller STC89C58RD+. However, for our all masked implementations of Ext, we adopt masking scheme presented in Figure 1; for those of PRG, we adopt masking scheme in [19].

In our present context, one central question to answer is to determine the target operations (which will subsequently determine the interesting intermediate) for side-channel adversary. For this purpose, one generally selects those operations where one known input and another secret key are mixed. For our implementations of Ext(resp. PRG), this corresponds to the bitwise AND (resp. S-box) operation. As a result, for the cases of Ext(resp. PRG), we choose the output of AND(resp. S-box) operation to be our interesting intermediate. In addition, we will consider a 8-bit key guess, which is exactly the word size of our hardware platform STC89C58RD+.

[6] 41.6880 equals the product of code length of HW and 16 (bytes).

5.1. Theoretical Evaluation

For theoretical security evaluation of concrete cryptographic instantiations, we use Mutual Information (MI for short) of any interesting intermediate as one quantitative metric to verify our analysis results in Section 3. For the computation of MI, we use histogram estimation method [17], with the number of bins being 9 [18]. In this way, we compute the MI for different implementations of Ext(resp. RPG) to show their leakage resilience accordingly. For our masked implementations of both Ext and PRG, the masking order $d = 1$. For our shuffled implementations of Ext, the shuffling degree $N1 = 24$, while it is $N2 = 16$ for PRG.The results are shown in Table 2.

Table 2: Mutual Information of Implementations of PRG (S-box) and Ext (AND)

Implementations	Ext (AND)	PRG (S-box)
Unprotected	0.7729	1.3640
Masked, $d = 1$	0.0348	0.1046
Shuffled	0.0459 ($N1 = 24$)	0.0788 ($N2 = 16$)
Masked and Shuffled, $d = 1$	0.0227 ($N1 = 24$)	0.0375 ($N2 = 16$)

Table 2 briefly shows how and to what extend typical countermeasures applied to Ext(resp. PRG) has hardened the practical leakage resilience. It can be seen from Table 2 that, for both Ext and PRG, applications of masking and shuffling do decrease the MI of the target intermediates, and hence decrease the leakages from the corresponding implementations. And among our considered four implementations of Ext(resp. PRG), application of combination of masking and shuffling increase the leakage resilience most. In addition, the MI of each one of our four implementations of Ext is always less than that of the corresponding implementation of PRG. This does justify the observation made in [14] that the elementary operations of Ext are "less informative" than the ones of the AES. Note, however, that masking outperforms shuffling in terms of increasing the practical leakage resilience for the case of Ext, while shuffling is better than masking for the case of PRG. We guess this may due to the fact that the value of x randomly chosen in our experiments has a large impact on the MI of subsequent AND operations(see in Figure 1), which gave rise to partial independence of the MI in some cases on y.

Another important aspect concerning the results worth noting is that the MI of shuffled implementation of Ext is less than that of unprotected imple-

mentation of Ext. Contrary to the case of MI, the min-entropy of the shuffled implementation of Ext is the same as that of unprotected implementation of Ext(see in Table 1 of Section 3). This fact, in some sense, evidently highlights some gaps between theoretically leakage resilient constructions and specific implementations of them.

5.2. Practical Evaluation

For practical security evaluation of concrete cryptographic instantiations, we perform template based DPA attacks against multiple implementations of Ext(resp. PRG). In this paper, we choose template based DPA attacks instead of other high-order DPA variants, because the former is widely believed to be the most powerful one and nearly always outperform the latter [13]. The number of power traces used for building template for each case of our experiments is shown in Table 3.

For masked implementation of Ext(resp. PRG), we consider *1-order* masking only, i.e. $d = 1$. For shuffled implementations of Ext, we consider shuffling degree $N1 = 6, 12, 24$, respectively; for those of PRG, we consider shuffling degree $N2 = 4, 8, 16$, respectively.

In our template building and attacks, the noise standard deviation is set to be $\sigma = 0.5$. And Success Rate (SR for short)[7], one quantitative security metric, is adopted to measure the practical leakage resilience of cryptographic implementations.

Table 3: Number of Traces Used for Building Templates

Implementations	Ext	PRG
Unprotected	20,000	20,000
Masked, $d = 1$	81,000	81,000
Shuffled	20,000, $N1 = 6$	20,000, $N2 = 4$
	40,000, $N1 = 12$	20,000, $N2 = 8$
	40,000, $N1 = 24$	20,000, $N2 = 16$
Masked and Shuffled, $d = 1$	81,000, $N1 = 24$	81,000, $N2 = 16$

[7] As a rule of thumb, high success rate usually implies low guessing entropy; so we don't consider guessing entropy in this work.

The sketch of our template building and attacks is as follows. For unprotected implementation of Ext(resp. PRG), we build template for each interesting intermediate. For masked implementation of Ext(resp. PRG), we build template for each pair of interesting intermediate and mask. During attacks, one tries to match each samples trace with the template, and the key guess that gives rise to the maximum matching probability will be treated as the correct key[9]. For shuffled implementation of Ext(resp. PRG), we build template for each interesting intermediate. During attacks, one tries to match the template with each of the N power consumptions of the target intermediate values that has been shuffled, and the sum of these N results is viewed as the template matching probability for one trace. In this way, the key guess that gives rise to the maximum matching probability will be treated as the correct key. For masked and shuffled implementation of Ext(resp. PRG), we build template for each pair of interesting intermediate and mask too. The template matching phase is identical with that of attacking shuffled implementation of Ext(resp. PRG).

5.2.1. Comparison of Implementations of Ext

We performed template based attacks against four implementations of Ext, with the target operation being bitwise AND. The results are presented in Figure 2. The SR of our attacks against masked and shuffled implementation of Ext is nearly zero, so it does not appear in Figure 2. Note that one power trace of Ext actually contains 128 AND operations, and therefore $1{,}536 (= 128 \times 12)$ power traces are used to attack the masked and shuffled implementation of Ext.

As explained in Section 3.4, the min-entropy of shuffled implementation of Ext is identical with that of unprotected implementation of Ext. However, in real world, it is harder to attack shuffled implementation than to attack unprotected implementation. Our experiments justify this intuition, because the min-entropy considers the worst-case while average-case is usually taken into account in practice. That is, shuffling helps increase the practical leakage resilience of cryptographic implementations, if properly used. These results, in some sense, again highlights the possible gaps between theoretically leakage resilient constructions and specific implementations of them. In addition, of our four implementations, the masked and shuffled implementation of Ext is the most secure in terms of SR, in the presence of template based DPA attacks.

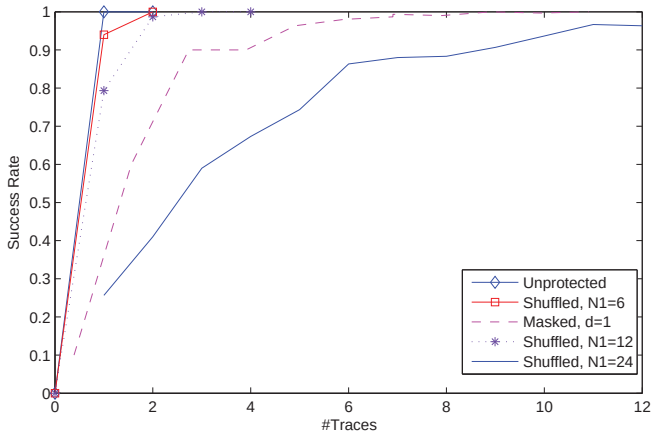

Figure 2: Success Rates of Attacks against Implementations of Ext

5.2.2. Comparison of Leakage Resilience of Implementations of PRG and Ext

In oder to justify whether or not the extractor is still the weaker point of the LR-SC scheme in protected implementations, we perform template based DPA attacks against our four implementations of Ext and those of PRG respectively. The results are presented in Figure 3. Similarly, the results of our attacks against the masked and shuffled implementation of Ext do not appear in 3, as the corresponding SR of these attacks are nearly zero.

The number of power traces for the SR of attacks against the masked implementation of Ext to reach 1 is 9, while it is 50 for that of PRG. In contrast, in the case of $N1 = 24$ and $N2 = 16$, the number of power traces for the SR of attacks against the shuffled implementation of Ext to reach 1 is only 13, while it is 55 for that of PRG. Therefore, for masked implementation of LR-SC, the extractor is still a better target for DPA attacks than an AES-based PRG. And it is also the same with the shuffled implementation of LR-SC. These results are consistent with the observations made in [14],[10].

On the other hand, and more importantly, in the case of $d = 1$, $N1 = 24$ and $N2 = 16$, 235 traces are sufficient enough for one to successfully attack the masked and shuffled implementation of PRG, while the SR of attacking the masked and shuffled implementation of Ext is nearly zero. This evidently shows that in this case, the extractor is no more a better target for DPA attacks than an AES-based PRG. That is, in this case, one should better attack the PRG other than the extractor. This observation

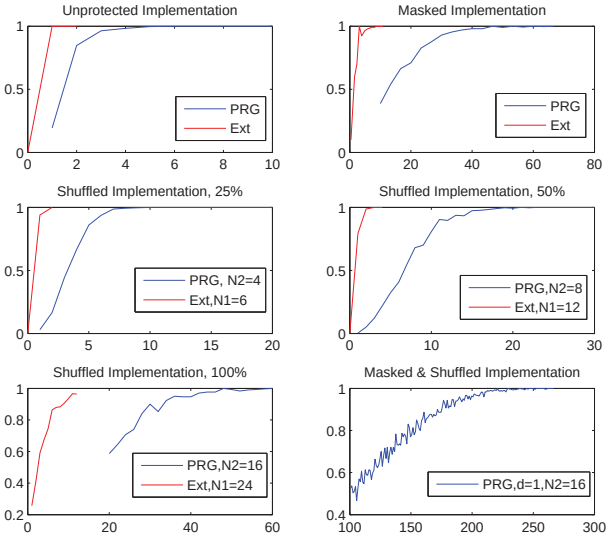

Figure 3: Success Rate of Attacks against Implementations of Ext and PRG

do contrast with previous ones[14],[10], which explicitly indicates that those existing observations are (mostly likely) incomplete.

6. Conclusions

Leakage resilient cryptography stands for one of important on-going efforts within world-wide cryptology communities towards building sound theoretical foundations of side-channel security of cryptography systems. Nonetheless, leakage resilient cryptography is still in its infancy. A number of concerns on leakage resilient cryptographic primitives need to be addressed. For example, to what extent these theoretical constructions relate to the physical reality. The possible practical significance of discussing this problem, among others, is to reveal the possible gaps (if any) between meaningful physical assumptions and sound theoretical proofs, thus unveiling the physical leakage resilience these theoretical constructs could provide.

Taking the example of random extractors, which have been proven to be fundamental building blocks for constructing leakage resilient cryptographic primitives, this paper investigates the feasibility of applying classical countermeasures to ameliorate the leakage resilience of cryptographic components

and/or primitives against side-channel attacks and show how to evaluate the physical leakage resilience theoretically and practically. The countermeasures we consider are masking, shuffling, and combinations of them. Even though our theoretical and experimental results justified some existing observations, they evidently and firmly reveal some new observations that contrast with knowing ones, which explicitly indicates that previous observations are (mostly likely) incomplete.

We argue that our work is of both obvious theoretical interest and important practical significance, and may help foster the further research on the design and implementation of random extractors in leakage-resilient cryptographic constructions. Such efforts could open the way towards more in-depth and more rigorous treatment of implementations and constructions of physically observable cryptographic schemes.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No.61073178) and Beijing Natural Science Foundation (No.4112064).

References

[1] Z. Brakerski, Y. Kalai, J. Katz, V. Vaikuntanathan. Overcoming the Hole in the Bucket: Public-Key Cryptography Resilient to Continual Memory Leakage.IEEE FOCS 2010, pp.501-510, 2010.

[2] Y. Dodis, A. Elbaz, R. Oliveira, R. Raz. Improved Randomness Extraction from Two Independent Sources.APPROX and RANDOM 2004, LNCS 3122, pp.334-344, 2004.

[3] Y. Dodis, K. Haralambiev, A. Lopez-Alt, D.Wichs. Cryptography Against Continuous Memory Attacks. IEEE FOCS 2010, pp.511-520, 2010.

[4] S. Dziembowski, K. Pietrzak. Leakage-Resilient Cryptography. IEEE FOCS 2008, pp.293-302, 2008.

[5] K. Gandolfi, C. Mourtel, F. Olivier. Electromagnetic Analysis: Concrete Results. CHES 2001, LNCS 2162, pp.251-261, 2001.

[6] P. C. Kocher, J. Jaffe, B. Jun. Differential Power Analysis. CRYPTO 1999, LNCS 1666, pp.388-397, 1999.

[7] P. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. CRYPTO 1996, LNCS 1109, pp.104-113, 1996.

[8] S. Halevi, H. Lin. After-the-Fact Leakage in Public-Key Encryption. TCC 2011, LNCS 6597, pp.107-124, 2011.

[9] S. Mangard, E. Oswald, T. Popp. Power Analysis Attacks Revealing the Secrets of Smart Cards.Springer-Verlag Press, 2007.

[10] M. Medwed, F. Standaert. Extractors Against Side-Channel Attacks: Weak or Strong? Journal of Cryptographic Engineering, Vol.1, No.3, pp.231-241, 2011.

[11] T. Malkin, I. Teeranishi, Y. Vahlis, M, Yung. Signatures Resilient to Continual Leakage on Memory and Computation.TCC 2011, LNCS 6597, pp.89-106, 2011.

[12] S. Moriai, S. Vaudenary. On the Pseudorandom of Top-Level Schemes of Block Ciphers. ASIACRYPT 2000, LNCS 1976, pp.280-302, 2000.

[13] E. Oswald, S. Mangard. Template Attacks on Masking-Resistance Is Futile. CT-RSA 2007, LNCS 4377, pp.243-256, 2007.

[14] F. X. Standaert. How Leaky Is an Extractor? LATINCRYPT 2010, LNCS 6212, pp.294-304, 2010.

[15] F. X. Standaert, T. G. Malkin, M. Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. EUROCRYPT 2009, LNCS 5497, pp.443-461, 2009.

[16] U. Vazirani. Efficiency Considerations in Using Semi-Random Sources. ACM STOC 1987, pp.160-168, 1987.

[17] B. Gierlichs, L. Batina, P. Tuyls, B. Preneel. Mutual Information Analysis A Generic Side-Channel Distinguisher. CHES 2008, LNCS 5154, pp. 426-442, 2008.

[18] N. V. Charvillon, F. X. Standaert. Generic Side-Channel Distinguishers: Improvements and Limitations. CRYPTO 2011, LNCS 6841, pp.354-372, 2011.

[19] C. Herbst, E. Oswald, S. Mangard. An AES Smart Card Implementation Resistant to Power Analysis Attacks. ACNS 2006, LNCS 3989, pp.239-252, 2006.

[20] S. Micali, L. Reyzin. Physically Observable Cryptography (Extended Abstract). TCC 2004, LNCS 2951, pp. 278-296, 2004.

[21] Y. Yu, F. X. Standaert, O. Pereira, M. Yung. Practical Leakage-Resilient Pseudorandom Generators. ACM CCS 2010, pp. 141-151, 2010.