

Title	Non-interactive Opening for Ciphertexts Encrypted by Shared Keys
Author(s)	Chen, Jiageng; Emura, Keita; Miyaji, Atsuko
Citation	Lecture Notes in Computer Science, 7043/2011: 57-68
Issue Date	2011-11-01
Type	Journal Article
Text version	author
URL	<a href="http://hdl.handle.net/10119/10297">http://hdl.handle.net/10119/10297</a>
Rights	This is the author-created version of Springer, Jiageng Chen, Keita Emura, and Atsuko Miyaji, Lecture Notes in Computer Science, 7043/2011, 2011, 57-68. The original publication is available at <a href="http://www.springerlink.com">www.springerlink.com</a> , <a href="http://dx.doi.org/10.1007/978-3-642-25243-3_5">http://dx.doi.org/10.1007/978-3-642-25243-3_5</a>
Description	



# Non-interactive Opening for Ciphertexts Encrypted by Shared Keys

Jiageng Chen<sup>1</sup> \*, Keita Emura<sup>2</sup> \*\*, and Atsuko Miyaji<sup>1</sup> \*\*\*

<sup>1</sup> School of Information Science, Japan Advanced Institute of Science and Technology, 1-1, Asahidai, Nomi, Ishikawa, 923-1292, Japan

<sup>2</sup> Center for Highly Dependable Embedded Systems Technology, JAIST, Japan  
{jg-chen,k-emura,miyaji}@jaist.ac.jp,

**Abstract.** Let a sender Alice computes a ciphertext  $C$  of a message  $M$  by using a receiver Bob's public key  $pk_B$ . Damgård, Hofheinz, Kiltz, and Thorbek (CT-RSA2008) has proposed the notion *public key encryption with non-interactive opening* (PKENO), where Bob can make a non-interactive proof  $\pi$  that proves the decryption result of  $C$  under  $sk_B$  is  $M$ , without revealing  $sk_B$  itself. When Bob would like to prove the correctness of  $(C, M)$  (e.g., the information  $M$  sent to Bob is not the expected one), PKENO turns out to be an effective cryptographic primitive. A PKENO scheme for the KEM/DEM framework has also been proposed by Galindo (CT-RSA2009). Bob can make a non-interactive proof  $\pi$  that proves the decapsulation result of  $C$  under  $sk_B$  is  $K$  without revealing  $sk_B$  itself, where  $K$  is an encapsulation key of the DEM part. That is, no verifier can verify  $\pi$  without knowing  $K$ . This setting is acceptable if  $K$  is an ephemeral value. However, PKENO is not applicable if an encryption key is shared among certain users beforehand, and is used for a relatively long period before re-running the key agreement protocol, such as symmetric cryptosystems. In this paper, we define the notion *secret key encryption with non-interactive opening* (SKENO), and give a generic construction of SKENO from verifiable random function (VRF) and the Berbain-Gilbert IV-dependent stream cipher construction (FSE2007). Bob can make a non-interactive proof  $\pi$  that proves the decryption result of  $C$  under  $K$  is  $M$ , without revealing  $K$  itself.

## 1 Introduction

### 1.1 PKENO: Public Key Encryption with Non-interactive Opening

Let's first consider the following scenario. Assuming that a sender Alice computes a ciphertext  $C$  of a message  $M$  by using a receiver Bob's public key  $pk_B$ . In order to solve the dispute in some circumstances, Bob would like to prove

---

\* This author is supported by the Graduate Research Program.

\*\* This author is supported by the Center for Highly Dependable Embedded Systems Technology as a Postdoc researcher.

\*\*\* This work is supported by Grant-in-Aid for Scientific Research (B), 20300003.

the correctness of the corresponding plaintext-ciphertext pair  $(C, M)$  (e.g., the information  $M$  sent to Bob is not the expected one). The easiest way to prove it is that Bob opens his secret key  $sk_B$ , demonstrates the decryption algorithm, and shows the decryption result of  $C$  under  $sk_B$  is  $M$ . However, no other ciphertext encrypted  $pk_B$  remains secure. To capture this situation above, Damgård, Hofheinz, Kiltz, and Thorbek [13] has proposed the notion *public key encryption with non-interactive opening* (PKENO), where Bob can make a non-interactive proof  $\pi$  that proves the decryption result of  $C$  under  $sk_B$  is  $M$ , without revealing  $sk_B$  itself. They also show a generic construction of PKENO from identity-based encryption (IBE) [9] and strongly existentially unforgeable one-time signature (OTS) [5]. Other generic construction of PKENO from group signature secure in the BSZ model [4] has been proposed [17], and concrete constructions of PKENO also have been proposed [20, 21, 28].

The main idea of PKENO construction is to make a decryption key which works about the corresponding ciphertext only, and the decryption key is set as a proof. For the sake of readability, we introduce the generic construction of PKENO (based on IBE and OTS) proposed by Damgård, Hofheinz, Kiltz, and Thorbek [13] as follows: Bob runs  $(pk_B, sk_B) \leftarrow \text{IBE.KeyGen}(1^k)$ , i.e.,  $sk_B$  is a master key of the underlying IBE scheme. Alice runs  $(vk, sk) \leftarrow \text{OTS.KeyGen}(1^k)$ , and computes  $C \leftarrow \text{IBE.Enc}(pk_B, vk, M)$ , i.e., a verification key  $vk$  is regarded as the identity of the underlying IBE scheme, and computes  $\sigma \leftarrow \text{OTS.Sign}(sk, C)$ . The PKENO ciphertext is  $(C, \sigma, vk)$ . Bob can decrypt  $(C, \sigma, vk)$  such that Bob verifies  $1 = \text{OTS.Verify}(vk, \sigma, C)$ , and computes  $usk[vk] \leftarrow \text{IBE.Extract}(sk_B, vk)$  and  $M \leftarrow \text{IBE.Dec}(usk[vk], C)$ . Then,  $usk[vk]$  can be set as  $\pi$ , since anyone can prove whether  $M = \text{IBE.Dec}(\pi, C)$  or not. Other ciphertexts encrypted by  $pk_B$  remain secure assuming that different  $vk$  is chosen in each encryption (this is a reasonable assumption of OTS).

## 1.2 PKENO with the KEM/DEM framework and its Limitation

A PKENO scheme for the KEM/DEM (Key Encapsulation/Decapsulation Mechanism) framework also have been proposed by Galindo [20]. To encrypt  $M$ , Alice compute  $(K, C_1) \leftarrow \text{Encapsulation}(pk_B)$  and  $C_2 \leftarrow \text{DEM}(K, M)$ , where DEM is a symmetric cipher, and sends  $C = (C_1, C_2)$  to Bob. Bob can compute  $K \leftarrow \text{Decapsulation}(sk_B, C_1)$  and  $M \leftarrow \text{DEM}(K, C_2)$ . In the Galindo's scheme, Bob can make a non-interactive proof  $\pi$  that proves the decapsulation result of  $C_1$  under  $sk_B$  is  $K$  without revealing  $sk_B$  itself. The construction methodology of the Galindo PKENO scheme with the KEM/DEM framework is also same as that of the previous PKENO scheme, where for a proof  $\pi$  anyone can compute  $K' \leftarrow \text{Decapsulation}(\pi, C_1)$  and check whether  $K = K'$  or not. That is, no verifier can verify  $\pi$  without knowing  $K$ . This setting is acceptable if  $K$  is an ephemeral value that changes per session (e.g., in the KEM/DEM usage). However, in some other applications where the secret key  $K$  is negotiated beforehand among a group of people, and it is then used by symmetric cryptosystems to do encryption for relatively a period of time (especially this is the case for IV-dependent stream cipher where the  $K$  is remained unchanged for many sessions

while IV is changed for each session and sent in plaintext form for synchronization), then the Galindo scheme does not work since the opening of  $K$  will expose the unrelated messages as well. Here, we make it clear that the situation where a ciphertext is encrypted by a shared key as follows.

- Let  $K$  be a shared key of Alice and Bob, and Charlie does not have  $K$ .
- Alice sends  $C$  to Bob, where  $C$  is a ciphertext of a message  $M$  under the key  $K$ .
- For Charlie, Bob wants to prove that the decryption result of  $C$  is  $M$  *without revealing*  $K$ .

PKENO is not capable to handle the situation above even a PKENO scheme with the KEM/DEM framework has been proposed. Note that the proof itself is the decryption key in the previous PKENO schemes. Since opening the shared key  $K$  is not an option anymore,  $K$  should not be recognized as a proof. Thus, we need to investigate a new methodology to handle the situation that a ciphertext is encrypted by a shared key which cannot be revealed.

### 1.3 Our Contribution

In this paper, we define the notion *secret key encryption with non-interactive opening* (SKENO), where

- Bob can make a non-interactive proof  $\pi$  that proves the decryption result of  $C$  under  $K$  is  $M$ , without revealing  $K$  itself,

and give a generic construction of SKENO from verifiable random function (VRF) [2, 14, 15, 26, 30, 33] and the Berbain-Gilbert IV-dependent stream cipher construction [6], where IV is an initial vector.

In the Berbain-Gilbert construction, pseudo-random function (PRF) [23] is regarded as a key scheduling algorithm (KSA) and a pseudo-random number generator (PRNG) [8, 36] is regarded as a pseudo-random generation algorithm (PRGA). From an  $n$ -bit initial vector IV and a  $k$ -bit secret key  $K$ , PRF (say  $F_K(IV)$ ) outputs an  $m$ -bit initial state (say  $y$ ) which is used as an input of PRNG. Finally, PRNG (say  $G(y)$ ) outputs  $L$ -bit keystream  $Z_{IV}$ . A ciphertext  $C$  is  $Z_{IV} \oplus M$ , where  $\oplus$  is the exclusive-or operation. Berbain and Gilbert give a composition theorem where the composition  $G \circ F_K$  is also PRF such that  $G \circ F : \{0, 1\}^n \rightarrow \{0, 1\}^L$ . Therefore,  $G \circ F$  can be a secure stream cipher, where no adversary  $\mathcal{A}$  can distinguish whether  $G(F_K(IV))$  is a truly random number or not, even if  $\mathcal{A}$  can select IV.

A VRF is a PRF that provides a non-interactively verifiable proof: given an input value  $x$  and its output  $y = F_{SK}(x)$ , a proof  $\pi(x)$  proves that  $y$  is the output of the function  $F$  indexed by  $SK$  given input  $x$ , without revealing a secret key  $SK$  itself. Several applications of VRF have been considered, e.g., non-interactive lottery system used in micropayment schemes [35], resettable zero-knowledge proofs [34], updatable zero knowledge databases [29], set intersection

protocols [25], compact e-cash [12, 3], adaptive oblivious transfer protocols [27], keyword search [19], and so on. We make it clear that the usage of VRF for the PKENO functionality has not been appeared to the best of our knowledge.

We set  $IV$  as an input of VRF (as in the Berbain-Gilbert construction) and set a shared key  $K$  as a secret key of VRF. VRF leads to an  $m$ -bit initial state which is the input to the PRNG for generating a  $L$ -bit keystream  $Z_{IV}$ . A ciphertext  $C$  is  $Z_{IV} \oplus M$ . We set  $\pi := (\pi', y = F_K(IV))$  a proof of  $(IV, C, M)$ , where  $K := SK$ . Due to the VRF functionality, one can prove that  $m$ -bit initial state is the result of the underlying VRF without revealing  $K$ .

Although we have to mention that the VRF primitive is a relatively expensive assumption, our construction can be considered to be efficient in the environments where the key and  $IV$  setup phase is not executed frequently, since the execution of PRNG will play a dominant role in the encryption and decryption process.

## 2 Preliminaries

In this section, we define PRNG and VRF. We denote *State* as the state information transmitted by the adversary to himself across stages of the attack in experiments.

### 2.1 PRNG: Pseudo-Random Number Generator

Pseudo-random number generator (PRNG)  $G : \{0, 1\}^m \rightarrow \{0, 1\}^L$  is used to expand an  $m$ -bit secret seed into an  $L$ -bit sequence ( $m < L$ ).

**Definition 1 (Pseudorandomness of PRNG).** *We say that a function  $G : \{0, 1\}^m \rightarrow \{0, 1\}^L$  is PRNG if for all probabilistic polynomial-time (PPT) adversary  $\mathcal{A}$ , the following advantage is a negligible function of the security parameter  $\lambda$ .*

$$\text{Adv}_{G, \mathcal{A}}^{\text{Pseudo}}(1^\lambda) := |\Pr[y^* \xleftarrow{\$} \{0, 1\}^m; b \xleftarrow{\$} \{0, 1\}; \\ Z_0^* \leftarrow G(y^*); Z_1^* \xleftarrow{\$} \{0, 1\}^L; b' \leftarrow \mathcal{A}(1^\lambda, Z_b^*); b = b'] - \frac{1}{2}|$$

As a well-known result, PRNGs exist if and only if one-way functions exist [24]. Or rather, from a 1-1 one-way function  $f$  (i.e., one-way permutations, OWPs), a PRNG  $G$  is easily constructed [22] such that  $G(y) = b(y)b(f(y))b(f^2(y)) \cdots b(f^{L-1}(y))$ , where  $b$  is a corresponding hard-core.

### 2.2 VRF: Verifiable Random Function

Verifiable random functions (VRFs) were proposed by Micali, Rabin, and Vadhan [33], and many VRF constructions have been proposed by applying the complexity assumptions used in public-key encryption constructions (e.g., the

strong RSA assumption [33], the  $q$ -Diffie-Hellman inversion assumption and the  $q$ -decisional bilinear Diffie-Hellman inversion assumption [15], the generalized Diffie-Hellman assumption [14], the  $\ell$ -Decisional Diffie-Hellman Exponent assumption [26] and so on). Some black-box separations/constructions also have been shown by Brakerski, Goldwasser, Rothblum, and Vaikuntanathan [11], where VRF (both weak one and standard one) cannot be constructed from OWPs in a black-box manner, and the existence of weak VRF is essentially equivalent to the existence of non-interactive zero-knowledge proofs for all NP languages in the common random string model. Since OWPs are sufficient for constructing most of basic symmetric-key primitives, which include, e.g., PRNG, PRF, symmetric-key encryption schemes, and message authentication codes (Matsuda et al. [32] provide a nice summary of OWPs), VRF is a strong tool for constructing a stream cipher (Fiore and Schröder explained feasibility of VRF [18] in details). However, for handling the PKENO functionality in the symmetric cryptosystems such strong cryptographic assumption is required. It is an interesting open problem to clarify whether a black-box separation (or construction) of SKENO based on a weaker primitive than VRF exists or not.

Next, we define VRF by referring the Hohenberger-Waters VRF definition [26] as follows. Let  $F : \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^m$ , where  $k = k(\lambda)$ ,  $n = n(\lambda)$ , and  $m = m(\lambda)$  are polynomials in the security parameter  $1^\lambda$ , be an efficient computable function. For all  $SK \in \{0, 1\}^k$ , simply we denote  $F_{SK} : \{0, 1\}^n \rightarrow \{0, 1\}^m$ .

**Definition 2 (VRF [26]).** *We say that  $F$  is a VRF if there exist algorithms (VRF.Setup, VRF.Prove, VRF.Verify) such that*

VRF.Setup( $1^\lambda$ ) *outputs a pair of keys*  $(PK, SK)$ ;

VRF.Prove( $SK, x$ ) *outputs a pair*  $(y, \pi')$ , *where*  $y = F_{SK}(x)$  *is the function value of*  $x \in \{0, 1\}^n$  *and*  $\pi'$  *is the proof of correctness; and*

VRF.Verify( $PK, x, y, \pi'$ ) *outputs 1 if*  $y = F_{SK}(x)$ , *and 0 otherwise.*

*In addition, three security notions are required:*

**Provability:** *This guarantees that an honestly generated proof is always accepted by the Verify algorithm, i.e., for all*  $(PK, SK) \leftarrow \text{VRF.Setup}(1^\lambda)$  *and*  $x \in \{0, 1\}^n$ , *if*  $(y, \pi') \leftarrow \text{VRF.Prove}(SK, x)$ , *then*  $\text{VRF.Verify}(PK, x, y, \pi') = 1$ .

**Uniqueness:** *This guarantees that no proof is accepted for different values*  $y_1 \neq y_2$  *and a common*  $x$ , *i.e., for all*  $(PK, SK) \leftarrow \text{VRF.Setup}(1^\lambda)$  *and*  $x \in \{0, 1\}^n$ , *there does not exist a tuple*  $(y_1, y_2, \pi'_1, \pi'_2)$  *such that*  $y_1 \neq y_2$  *and*  $\text{VRF.Verify}(PK, x, y_1, \pi'_1) = \text{VRF.Verify}(PK, x, y_2, \pi'_2) = 1$ . *Note that no adversary can break uniqueness even*  $SK$  *is opened since such tuple does not exist. This property is used for realizing the proof soundness property (which is defined in Section 3).*

**Pseudorandomness:** *This guarantees that no adversary can distinguish whether an output of*  $F$  *is truly random or not, i.e., for all PPT adversary*  $\mathcal{A}$ , *the following advantage is a negligible function of the security parameter*  $\lambda$ .

$$\begin{aligned}
\text{Adv}_{F, \mathcal{A}}^{\text{Pseudo}}(1^\lambda) &:= \\
&|\Pr[(PK, SK) \leftarrow \text{VRF.Setup}(1^\lambda); (x^*, \text{State}) \leftarrow \mathcal{A}^{\text{VRF.Prove}(SK, \cdot)}(1^\lambda, PK); \\
&b \xleftarrow{\$} \{0, 1\}; y_0^* \leftarrow F_{SK}(x^*); y_1^* \xleftarrow{\$} \{0, 1\}^m; b' \leftarrow \mathcal{A}^{\text{VRF.Prove}(SK, \cdot)}(y_b^*, \text{State}); \\
&b = b'] - \frac{1}{2}|
\end{aligned}$$

where  $\text{VRF.Prove}(SK, \cdot)$  is the prove oracle which takes as input  $x \in \{0, 1\}^n$  ( $x \neq x^*$ ), outputs  $(F_{SK}(x), \pi')$ .

**Remark:** As a similar cryptographic primitive of VRF, verifiable pseudo-random bit generators (VPRGs) has been introduced by Dwork and Naor [16], where the holder of the seed can generate proofs of consistency for some parts of the sequence without hurting the unpredictability of the remaining bits. Note that in our SKENO construction we do not have to use VPRG (instead of PRNG) by assuming that the underlying PRNG satisfies the soundness property, where no adversary can find  $(y_1, y_2) \in \{0, 1\}^m \times \{0, 1\}^m$  such that  $G(y_1) = G(y_2)$  and  $y_1 \neq y_2$ . Note that this requirement is natural, e.g., Bertoni, Daemen, Peeters, and Assche [7] have mentioned that “loading different seeds into the PRNG shall result in different output sequences. In this respect, a PRNG is similar to a cryptographic hash function that should be collision-resistant”.

### 3 SKENO: Secret Key Encryption with Non-interactive Opening

In this section, we give the definition of (IV-dependent) SKENO. We assume that each IV is randomly chosen for each encryption. A SKENO consists of five algorithms, KeyGen, Enc, Dec, Prove, and Verify.

**Definition 3 (System operation of SKENO).**

**KeyGen**( $1^\lambda$ ): This algorithm takes as inputs a security parameter  $\lambda \in \mathbb{N}$ , and returns a public verification key  $VK$  and a secret key  $K$ .

**Enc**( $K, IV, M$ ): This algorithm takes as inputs  $K$  and an initial vector  $IV \in \{0, 1\}^n$  and a message  $M \in \{0, 1\}^L$ , and returns a ciphertext  $C$ . We assume that  $IV$  is also sent to a decryptor.

**Dec**( $K, IV, C$ ): This algorithm takes as inputs  $K$  and  $C$ , and returns  $M$  or  $\perp$ .

**Prove**( $K, IV, C$ ): This algorithm takes as inputs  $K$ ,  $IV$ , and  $C$ , and returns a proof  $\pi$ .

**Verify**( $VK, IV, C, M, \pi$ ): This algorithm takes as inputs  $VK$ ,  $IV$ ,  $C$ ,  $M$ , and  $\pi$ , and returns 1 if  $C$  is the ciphertext of  $M$  and  $IV$  under  $K$ , and 0, otherwise.

We require correctness and completeness as follows.

**Definition 4 (Correctness).** For all  $(VK, K) \leftarrow \text{KeyGen}(1^\lambda)$ ,  $IV \in \{0, 1\}^n$ , and  $M \in \{0, 1\}^L$ ,  $\text{Dec}(K, IV, \text{Enc}(K, IV, M)) = M$  holds.

**Definition 5 (Completeness).** For all  $(VK, K) \leftarrow \text{KeyGen}(1^\lambda)$ ,  $IV \in \{0, 1\}^n$  and for any ciphertext  $C$ , we have that for  $M \leftarrow \text{Dec}(K, IV, C)$ ,  $\text{Verify}(VK, IV, C, M, \text{Prove}(K, IV, C)) = 1$  holds with overwhelming probability (note that  $M$  may be  $\perp$ ).

Next, we define security notions of SKENO, called indistinguishability under chosen-ciphertext and prove attacks (IND-CCPA) and proof soundness.

**Definition 6 (IND-CCPA).** We say that a SKENO scheme  $\Pi$  is IND-CCPA secure if for all PPT adversary  $\mathcal{A}$  the following advantage is negligible in the security parameter.

$$\begin{aligned} \text{Adv}_{\Pi, \mathcal{A}}^{\text{Ind-ccpa}}(1^\lambda) &:= |\Pr[(VK, K) \leftarrow \text{KeyGen}(1^\lambda); \\ &\quad (IV^*, M_0^*, M_1^*, \text{State}) \leftarrow \mathcal{A}^{\text{Enc}(K, \cdot, \cdot), \text{Dec}(K, \cdot, \cdot), \text{Prove}(K, \cdot, \cdot)}(1^\lambda, VK); \\ &\quad b \xleftarrow{\$} \{0, 1\}; C^* \leftarrow \text{Enc}(K, IV^*, M_b^*); \\ &\quad b' \leftarrow \mathcal{A}^{\text{Enc}(K, \cdot, \cdot), \text{Dec}(K, \cdot, \cdot), \text{Prove}(K, \cdot, \cdot)}(C^*, \text{State}); b = b'] - \frac{1}{2}| \end{aligned}$$

$\text{Enc}(K, \cdot, \cdot)$  is the encryption oracle which takes as input  $IV \in \{0, 1\}^n$  and  $M \in \{0, 1\}^L$ , where  $IV \neq IV^*$ , outputs  $C \leftarrow \text{Enc}(K, IV, M)$ .  $\text{Dec}(K, \cdot, \cdot)$  is the decryption oracle which takes as input  $IV \in \{0, 1\}^n$  and  $C \in \{0, 1\}^L$ , where  $(IV, C) \neq (IV^*, C^*)$ , outputs  $M/\perp \leftarrow \text{Dec}(K, IV, C)$ .  $\text{Prove}(K, \cdot, \cdot)$  is the the prove oracle which takes as input  $IV \in \{0, 1\}^n$  and  $C \in \{0, 1\}^L$ , where  $IV \neq IV^*$ , outputs  $\pi \leftarrow \text{Prove}(K, IV, C)$ .

Indistinguishability under chosen-plaintext and prove attacks (IND-CPPA) is simply defined by excluding the Dec oracle from the IND-CCPA definition.

**Definition 7 (Proof Soundness).** We say that a SKENO scheme  $\Pi$  satisfies proof soundness if for all PPT adversary  $\mathcal{A}$  the following advantage is negligible in the security parameter.

$$\begin{aligned} \text{Adv}_{\Pi, \mathcal{A}}^{\text{sound}}(1^\lambda) &:= \Pr[(VK, K) \leftarrow \text{KeyGen}(1^\lambda); (IV, M, \text{State}) \leftarrow \mathcal{A}(1^\lambda, VK, K); \\ &\quad C \leftarrow \text{Enc}(K, IV, M); (\tilde{M}, \tilde{\pi}) \leftarrow \mathcal{A}(C, \text{State}); \\ &\quad \text{Verify}(VK, IV, C, \tilde{M}, \tilde{\pi}) = 1; M \neq \tilde{M}] \end{aligned}$$

## 4 Our SKENO Construction

In this section, we give our IND-CPPA secure SKENO scheme. An IND-CCPA secure SKENO scheme will be covered in Section 6. Let  $(\text{VRF.Setup}, \text{VRF.Prove}, \text{VRF.Verify})$  be a VRF and  $G$  be a PRNG.



**Protocol 1 (Proposed IND-CPPA secure SKENO).**

**KeyGen**( $1^\lambda$ ): Run  $(PK, SK) \leftarrow \text{VRF.Setup}(1^\lambda)$ , and output  $(VK, K) = (PK, SK)$ .

**Enc**( $K, IV, M$ ): Run  $(y, \pi') \leftarrow \text{VRF.Prove}(K, IV)$  and  $Z_{IV} \leftarrow G(y)$ , compute  $C = M \oplus Z_{IV}$ , and output  $C$ .

**Dec**( $K, IV, C$ ): Run  $(y, \pi') \leftarrow \text{VRF.Prove}(K, IV)$  and  $Z_{IV} \leftarrow G(y)$ , compute  $M = C \oplus Z_{IV}$ , and output  $M$ .

**Prove**( $K, IV, C$ ): Run  $(y, \pi') \leftarrow \text{VRF.Prove}(K, IV)$  and return  $\pi = (y, \pi')$ .

**Verify**( $VK, IV, C, M, \pi$ ): Parse  $\pi = (y, \pi')$ . If  $\pi' \neq \perp$ ,  $\text{VRF.Verify}(VK, IV, y, \pi') = 1$ , and  $C \oplus G(y) = M$ , then output 1. Otherwise, output 0.

The correctness clearly holds. In addition, completeness also holds if the underlying VRF satisfies provability.

Note that in our IND-CPPA secure SKENO the Dec algorithm never outputs  $\perp$  (i.e., even if  $C$  was not generated by the Enc algorithm, the Dec algorithm outputs the corresponding decryption result which belongs to the plaintext space  $\{0, 1\}^L$ ). Therefore, the Prove algorithm does not have to prove that  $C$  is an invalid ciphertext in the IND-CPPA secure SKENO, whereas, in the IND-CCPA SKENO (presented in Section 6), since the Dec algorithm outputs  $\perp$  for invalid ciphertexts, the Prove algorithm needs to accept the proof that the decryption result of  $C$  is  $\perp$ .

## 5 Security Analysis

In this section, we give the security proofs of our SKENO construction.

**Theorem 1.** *Our SKENO construction is IND-CPPA secure if the underlying VRF and PRNG satisfy pseudorandomness.*

*Proof.* Let  $\mathcal{A}$  be an adversary who can break the IND-CPPA security of our SKENO construction. We construct an algorithm  $\mathcal{B}$  that breaks the pseudorandomness of the underlying VRF. Let  $\mathcal{C}_{\text{VRF}}$  be the challenger of the underlying VRF.  $\mathcal{C}_{\text{VRF}}$  sends  $PK$  to  $\mathcal{B}$ .  $\mathcal{B}$  sets  $VK = PK$  and sends  $VK$  to  $\mathcal{A}$ . To answer queries issued by  $\mathcal{A}$ ,  $\mathcal{B}$  manages a table  $\{(IV, y, \pi')\}$ .

### Phase 1 query

**Enc:** Let  $(IV, M)$  be an encryption query issued by  $\mathcal{A}$ . If there exists the entry  $(IV, y, \pi')$ , then  $\mathcal{B}$  returns  $C = M \oplus G(y)$ . Otherwise,  $\mathcal{B}$  sends  $IV$  to  $\mathcal{C}_{\text{VRF}}$  as a VRF.Prove query, obtains  $(y, \pi')$ , adds  $(IV, M, y, \pi')$  to the table, and returns  $C = M \oplus G(y)$ .

**Prove:** Let  $(IV, C)$  be a prove query issued by  $\mathcal{A}$ . If there exists the entry  $(IV, y, \pi')$ , then  $\mathcal{B}$  returns  $\pi = (y, \pi')$ . Otherwise (this means that  $C$  is not generated via the Enc oracle), then  $\mathcal{B}$  sends  $IV$  to  $\mathcal{C}_{\text{VRF}}$  as a VRF.Prove query, obtains  $(y, \pi')$ , adds  $(IV, y, \pi')$  to the table, where  $M = C \oplus G(y)$ , and returns  $\pi = (y, \pi')$ .

### Challenge phase

In the challenge phase,  $\mathcal{A}$  sends  $(IV^*, M_0^*, M_1^*)$  to  $\mathcal{B}$ , and  $\mathcal{B}$  sends  $IV^*$  to  $\mathcal{C}_{\text{VRF}}$ .  $\mathcal{C}_{\text{VRF}}$  selects  $b \in \{0, 1\}$  and computes  $y^*$  such that  $y^* = F_K(IV^*)$  if  $b = 0$ , and  $y^* \xleftarrow{\$} \{0, 1\}^m$  if  $b = 1$ , and sends  $y^*$  to  $\mathcal{B}$ .  $\mathcal{B}$  randomly chooses  $b'' \in \{0, 1\}$ , computes  $C^* = M_{b''}^* \oplus G(y^*)$ , and sends  $C^*$  to  $\mathcal{A}$ .

### Phase 2 query

**Enc:** The same as the phase 1 query since  $\mathcal{A}$  never issues  $IV^*$  as an encryption query.

**Prove:** Let  $(IV, C)$  be a prove query issued by  $\mathcal{A}$ , where  $IV \neq IV^*$ . If there exists the entry  $(IV, y, \pi')$ , then  $\mathcal{B}$  returns  $\pi = (y, \pi')$ . Otherwise (this means that  $C$  is not generated via the Enc oracle), then  $\mathcal{B}$  sends  $IV$  to  $\mathcal{C}_{\text{VRF}}$  as a VRF.Prove query, obtains  $(y, \pi')$ , adds  $(IV, y, \pi')$  to the table, where  $M = C \oplus G(y)$ , and returns  $\pi = (y, \pi')$ .

### Guessing phase

Finally,  $\mathcal{A}$  outputs  $b' \in \{0, 1\}$ . Note that if  $b = 0$  (i.e.,  $y^* = F_K(IV^*)$ ), then  $C^*$  is a valid ciphertext of  $M_{b''}^*$ . So,  $\mathcal{A}$  has an advantage. Otherwise, let  $b = 1$  (i.e.,  $y^*$  is a random value, and is independent of  $IV^*$ ). Then if there exist an algorithm that can distinguish the probabilistic distribution of  $M_0^* \oplus G(y^*)$  and the probabilistic distribution of  $M_1^* \oplus G(y^*)$ , then it contradicts the fact that  $G$  is a PRNG. So, these distributions are identical if  $G$  is a PRNG. So  $\mathcal{A}$  has no advantage in the case of  $b = 1$ . Therefore, if  $b' = b''$ , then  $\mathcal{B}$  outputs 0, and 1, otherwise.  $\square$

**Theorem 2.** *Our SKENO construction satisfies proof soundness if the underlying VRF satisfy uniqueness.*

*Proof.* Let  $(M, \tilde{M})$  is a message pair and  $C \leftarrow \text{Enc}(K, IV, M)$  (these are appeared in the proof soundness definition). Since  $M \neq \tilde{M}$ , for  $Z_{IV} := C \oplus M$  and  $\tilde{Z}_{IV} := C \oplus \tilde{M}$ ,  $Z_{IV} \neq \tilde{Z}_{IV}$  holds. In addition, there exist  $y, \tilde{y} \in \{0, 1\}^m$  such that  $Z_{IV} = G(y)$  and  $\tilde{Z}_{IV} = G(\tilde{y})$ . Note that  $y \neq \tilde{y}$  since  $Z_{IV} \neq \tilde{Z}_{IV}$  and  $G$  is a deterministic function. This never happen if the underlying VRF satisfy uniqueness.  $\square$

## 6 SKENO with Chosen-Ciphertext Security

As in the conversion from CPA-secure DEM to CCA-secure DEM using message authentication code (MAC) [1], we can construct a IND-CCPA secure SKENO scheme. We use strongly existentially unforgeable against a one-time chosen-message attack (one-time sEU-CMA) MAC [10, 31]. A MAC consists of two algorithms, MAC and Vrfy. The MAC algorithm takes as inputs a secret key  $Z \in \{0, 1\}^{L_2}$  and a message  $M \in \{0, 1\}^{L_1}$  ( $L_1, L_2 \in \mathbb{N}$ ), and outputs a tag  $t \leftarrow \text{MAC}_Z(M)$ . The Vrfy algorithm takes as inputs  $Z$  and  $t$ , and outputs 0 (reject) or 1 (accept). We require that for all  $Z$  and  $M$  we have  $\text{Vrfy}_Z(M, \text{MAC}_Z(M)) = 1$ .

Briefly,  $Z_{IV} = G(F_K(IV))$  is divided into two part, say  $Z_1 \in \{0,1\}^{L_1}$  and  $Z_2 \in \{0,1\}^{L_2}$ , where  $L_1 + L_2 = L$  and  $L_2$  is the key length of the underlying MAC algorithm (MAC,Vrfy). Then, a plaintext  $M \in \{0,1\}^{L_1}$  is encrypted by  $Z_1$  such that  $c = M \oplus Z_1$ , and a tag  $t$  is computed under the key  $Z_2$  such that  $t = \text{MAC}_{Z_2}(c)$ . The actual ciphertext is  $C = (c, t)$ . In the Dec algorithm, run  $(y, \pi') \leftarrow \text{VRF.Prove}(K, IV)$  and  $Z_{IV} \leftarrow G(y)$ , divide  $Z_{IV} = Z_1 || Z_2$ , compute  $M = c \oplus Z_1$ , and output  $M$  if  $\text{Vrfy}_{Z_2}(c) = 1$  holds, and  $\perp$ , otherwise. In the Prove algorithm, run  $(y, \pi') \leftarrow \text{VRF.Prove}(K, IV)$  and  $Z_{IV} \leftarrow G(y)$ , and divide  $Z_{IV} = Z_1 || Z_2$ . If  $\text{Vrfy}_{Z_2}(c) = 1$  holds, then set  $\pi = (y, \pi')$ . If  $\text{Vrfy}_{Z_2}(c) = 0$  holds, then set  $\pi = (y, \perp)$ . In the Verify algorithm, if  $\pi' = \perp$  and  $\text{Vrfy}_{Z_2}(c) = 0$ , then output 1. If  $\pi' \neq \perp$ ,  $\text{VRF.Verify}(VK, IV, y, \pi') = 1$ , and  $\text{Vrfy}_{Z_2}(c) = 1$ , then output 1. Otherwise, output 0.

## 7 Conclusion

The previous proposed Non-interactive opening ciphertexts techniques are for public key cryptosystem, which cannot bring the solutions for the situation where the symmetric key cryptosystem is being used, and the secret key is shared among a group of people, since the key itself is not appropriate for opening to behave as a proof. This paper fills in the above gaps by proposing the first stream cipher based SKENO scheme that can be proved to be IND-CPPA and IND-CCPA secure, which can provide Non-interactive opening services in the shared key environment.

## Acknowledgements

The authors would like to thank the anonymous reviewers of ICICS 2011 for their invaluable comments, and also would like to thank Dr. Takahiro Matsuda (AIST, Japan) for his invaluable comments to help to improve this paper.

## References

1. ISO CD 18033-2. Encryption algorithms part 2:asymmetric ciphers., 2004.
2. Michel Abdalla, Dario Catalano, and Dario Fiore. Verifiable random functions from identity-based key encapsulation. In *EUROCRYPT*, pages 554–571, 2009.
3. Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. Compact E-cash and simulatable VRFs revisited. In *Pairing*, pages 114–131, 2009.
4. Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In *CT-RSA*, pages 136–153, 2005.
5. Mihir Bellare and Sarah Shoup. Two-tier signatures, strongly unforgeable signatures, and fiat-shamir without random oracles. In *Public Key Cryptography*, pages 201–216, 2007.
6. Côme Berbain and Henri Gilbert. On the security of IV dependent stream ciphers. In *FSE*, pages 254–273, 2007.

7. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. Sponge-based pseudo-random number generators. In *CHES*, pages 33–47, 2010.
8. Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984.
9. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, pages 213–229, 2001.
10. Dan Boneh and Jonathan Katz. Improved efficiency for CCA-secure cryptosystems built using identity-based encryption. In *CT-RSA*, pages 87–103, 2005.
11. Zvika Brakerski, Shafi Goldwasser, Guy N. Rothblum, and Vinod Vaikuntanathan. Weak verifiable random functions. In *TCC*, pages 558–576, 2009.
12. Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact E-cash. In *EUROCRYPT*, pages 302–321, 2005.
13. Ivan Damgård, Dennis Hofheinz, Eike Kiltz, and Rune Thorbek. Public-key encryption with non-interactive opening. In *CT-RSA*, pages 239–255, 2008.
14. Yevgeniy Dodis. Efficient construction of (distributed) verifiable random functions. In *Public Key Cryptography*, pages 1–17, 2003.
15. Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In *Public Key Cryptography*, pages 416–431, 2005.
16. Cynthia Dwork and Moni Naor. Zaps and their applications. *SIAM J. Comput.*, 36(6):1513–1543, 2007.
17. Keita Emura, Goichiro Hanaoka, and Yusuke Sakai. Group signature implies PKE with non-interactive opening and threshold PKE. In *IWSEC*, pages 181–198, 2010.
18. Dario Fiore and Dominique Schröder. Uniqueness is a different story: Impossibility of verifiable random functions from trapdoor permutations. Cryptology ePrint Archive, Report 2010/648, 2010. <http://eprint.iacr.org/>.
19. Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In *TCC*, pages 303–324, 2005.
20. David Galindo. Breaking and repairing Damgård et al. public key encryption scheme with non-interactive opening. In *CT-RSA*, pages 389–398, 2009.
21. David Galindo, Benoît Libert, Marc Fischlin, Georg Fuchsbauer, Anja Lehmann, Mark Manulis, and Dominique Schröder. Public-key encryption with non-interactive opening: New constructions and stronger definitions. In *AFRICACRYPT*, pages 333–350, 2010.
22. Oded Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, New York, NY, USA, 2001.
23. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.
24. Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudo-random generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.
25. Carmit Hazay and Yehuda Lindell. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In *TCC*, pages 155–175, 2008.
26. Susan Hohenberger and Brent Waters. Constructing verifiable random functions with large input spaces. In *EUROCRYPT*, pages 656–672, 2010.
27. Stanislaw Jarecki and Xiaomin Liu. Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In *TCC*, pages 577–594, 2009.
28. Junzuo Lai, Robert H. Deng, Shengli Liu, and Weidong Kou. Efficient CCA-secure PKE from identity-based techniques. In *CT-RSA*, pages 132–147, 2010.

29. Moses Liskov. Updatable zero-knowledge databases. In *ASIACRYPT*, pages 174–198, 2005.
30. Anna Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In *CRYPTO*, pages 597–612, 2002.
31. Takahiro Matsuda, Goichiro Hanaoka, Kanta Matsuura, and Hideki Imai. An efficient encapsulation scheme from near collision resistant pseudorandom generators and its application to IBE-to-PKE transformations. In *CT-RSA*, pages 16–31, 2009.
32. Takahiro Matsuda and Kanta Matsuura. On black-box separations among injective one-way functions. In *TCC*, pages 597–614, 2011.
33. Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *FOCS*, pages 120–130, 1999.
34. Silvio Micali and Leonid Reyzin. Soundness in the public-key model. In *CRYPTO*, pages 542–565, 2001.
35. Silvio Micali and Ronald L. Rivest. Micropayments revisited. In *CT-RSA*, pages 149–163, 2002.
36. Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *FOCS*, pages 80–91, 1982.