SCIENCE CHINA
Information Sciences

• **Supplementary File** •

Verifiable Random Functions with Boolean Function Constraints

Qianwen WANG[1], Rongquan FENG[1] & Yan ZHU[2*]

[1]*School of Mathematical Sciences, Peking University, Beijing,100871, China;*
[2]*School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing, 100083, China*

Appendix A The Proof of CVRF-BF

According to the definition of CVRF, we prove that our CVRF-BF scheme satisfies the security properties, including conditional provability, uniqueness, and pseudorandomness, as follows.

Appendix A.1 Conditional Provability

At first, we prove the conditional provability of our CVRF-BF scheme in terms of the following theorem.

Theorem 1. CVRF - BF scheme we constructed is in line with the Conditional Provability, that is, for all $(pk, sk) \in Setup(1^\kappa, n)$, $pk_f \in GenFun(pk, f)$ and all $x \in \{0,1\}^{in(\kappa)}$, if $(y, z) = Prove(sk, x)$, there exists a negligible polynomial μ such that

$$\Pr[Verify(pk, x, y, z) = b | \exists b \in \{0,1\}, f(x) = b] > 1 - u(\kappa). \tag{A1}$$

Proof. Firstly, for a given input $x = (x_1, x_2, \cdots, x_n) \in \{0,1\}^n$, the randomized output y and its witness z can be generated effectively in terms of Equation (A22). However, the value (y, z) is independent of the boolean function $f(\cdot)$. For a given boolean function $f(\cdot)$, the verification result can be divided into two cases:

- Case $f(x) = 1$: In $\mathcal{S}_1$, the equations in $Verify$ are based on the facts as follows:

$$\begin{aligned} e(z_i, H' \cdot H^x) &= e(G^{\frac{1}{\xi+x}} \cdot G_{i,x_i}^{r_i}, H^{\xi+x}) \\ &= e(G, H) \cdot e(G_{i,x_i}^{r_i}, H^{\xi+x}) \\ &= e(G, H) \cdot e(G_{i,x_i}, H^{r_i(\xi+x)}) \\ &= e(G, H) \cdot e(G_{i,x_i}, H_i' \cdot H_i^x) \end{aligned} \tag{A2}$$

In $\mathcal{S}_2$, we canobviously see that

$$\begin{aligned} y_i &= e(G, H)^{\frac{g_k(v_i)}{\xi+x}} \\ &= e(G^{\frac{1}{\xi+x}}, H^{g_k(v_i)}) \cdot e(G_{i,x_i}^{r_i}, H^{g_k(v_i)}) \cdot e(G_{i,x_i}^{g_k(v_i)}, H^{r_i})^{-1} \\ &= e(G^{\frac{1}{\xi+x}} \cdot G_{i,x_i}^{r_i}, H^{g_k(v_i)}) \cdot e(G_{i,x_i}^{g_k(v_i)}, H^{r_i})^{-1} \\ &= e(z_i, \tilde{H}_i) \cdot e(\tilde{G}_{i,x_i}, H_i)^{-1}. \end{aligned} \tag{A3}$$

To verify whether $y = H(sk, x)$ was computed correctly, the algorithm sets Lagrangian interpolation coefficient $\gamma_i = \prod_{1 \leqslant j \leqslant n, j \neq i} \frac{v_j}{v_j - v_i} \pmod p$. Then it needs to check the equations

$$y = \prod_{i=1}^{n} y_i^{\gamma_i} = e(G, H)^{\frac{\xi}{\xi+x}}. \tag{A4}$$

Hence, if $f(x) = 1$, the equation above has been checked. So we have the successful probability of verification

$$\Pr[Verify(pk, x, y, z, w) = 1 | f(x) = 1] = 1. \tag{A5}$$

* Corresponding author (email: zhuyan@ustb.edu.cn)

- Case $f(x)=0$: By our construction, $f(x)=0$ means in $f(x)=\vee_{k=1}^{m} conj_k$, all $conj_k=0$. In our algorithm the calculation will be end, and outputs 0 and does not pass the verification. Now we will show that, though the adversary could pass the step-1, the step-2 of the verification will not be passed and output 0. Assuming that in the step-2 verification, for input x, we get output y and witness z.

Then we also use (pk,x,y,z) to check the equations in $Verify$ as follows:

$$\begin{aligned} e(z_i, H'\cdot H^x) &= e(G^{\frac{1}{\xi+x}}\cdot G_{i,x_i}^{r_i}, H^{\xi+x}) \\ &= e(G,H)\cdot e(G_{i,x_i}^{r_i}, H^{\xi+x}) \\ &= e(G,H)\cdot e(G_{i,x_i}, H^{r_i(\xi+x)}) \\ &= e(G,H)\cdot e(G_{i,x_i}, H_i'\cdot H_i^x) \end{aligned} \tag{A6}$$

The equation above will be passed. Then we use the valid $(z_1,z_2,\cdots,z_n)$, it makes use of the public key pk and the found index k, to compute the corresponding y_i by

$$y_i = e(z_i,\tilde{H}_i)\cdot e(\tilde{G}_{i,x_i},H_i)^{-1}. \tag{A7}$$

Then the algorithm sets Lagrangian interpolation coefficient $\gamma_i=\prod_{1\leqslant j\leqslant n, j\neq i}\frac{v_j}{v_j-v_i} \pmod p$. And it needs to check the equations

$$y=\prod_{i=1}^{n} y_i^{\gamma_i} = e(G,H)^{\frac{\xi}{\xi+x}} \in \mathbb{G}_T. \tag{A8}$$

And if $f(x)=0$, z could be computed, but verifier can not get the secret key ξ. Then the equation (A6) can be verified we need $\tilde{G}_{i,x_i}=G_{i,x_i}^{g_k(v_i)}$, but $g_k(X)$ only exists when $conj_k=1$. But $f(x)=0$ means in $f(x)=\vee_{k=1}^{m} conj_k$, all $conj_k=0$. So $G_{i,x_i}^{g_k(v_i)}$ does not exist for all $i\in\{1,2,\cdots,n\}$. Hence, the equation (A7) will not pass, so does the equation (A8). Therefore, $Verify(pk,x,y,z)=0$. This means

$$\Pr[Verify(pk,x,y,z)=0 | f(x)=0]=1. \tag{A9}$$

Above all, by the two equations (A5) and (A9), we know that

$$\Pr[Verify(pk,x,y,z)=b | \exists b\in\{0,1\}, f(x)=b] > 1-u(\kappa), \tag{A10}$$

where $u(\kappa)=0$. Hence, the theorem holds.

Appendix A.2 Uniqueness

Secondly, we have to prove the uniqueness of our CVRF-BF scheme in terms of the following theorem.

Theorem 2. CVRF - BF scheme we constructed complies with the definition of Uniqueness, that is, in the groups $\mathbb{G},\mathbb{G}_T$ of order p and a bilinear map $e:\mathbb{G}\times\mathbb{G}\to\mathbb{G}_T$, for all $(pk,sk)\in Setup(1^\kappa,n)$ and inputs $x\in\{0,1\}^{in(\kappa)}$, there does not exist a tuple $(y^{(1)},y^{(2)},z^{(1)},z^{(2)})$ such that

$$Pr[y^{(1)}\neq y^{(2)} | Verify(pk,x,y^{(1)},z^{(1)}=1, Verify(pk,x,y^{(2)},z^{(2)})=1] \leqslant \mu(\kappa). \tag{A11}$$

Proof. If there are different outputs and witnesses $y^{(1)}=(y_1^{(1)},y_2^{(1)},\cdots,y_n^{(1)})$ and $y^{(2)}=(y_1^{(2)},y_2^{(2)},\cdots,y_n^{(2)})$, $z^{(1)}=(z_1^{(1)},z_2^{(1)},\cdots,z_n^{(1)})$ and $z^{(2)}=(z_1^{(2)},z_2^{(2)},\cdots,z_n^{(2)})$ with the same inputs $x=(x_1,x_2,\cdots,x_n)$, and they both have $f(x)=1$ and $Verify(pk,x,y^{(1)},z^{(1)}=1$ and $Verify(pk,x,y^{(2)},z^{(2)})=1$ set up at the same time.

By our construction, in terms of $f(x)=\vee_{k=1}^{m} conj_k$ this algorithm finds a clause $conj_k$ to satisfy $conj_k=1$ for $x=(x_1,x_2,\cdots,x_n)$. Then they all pass the verify functions. So we have these formulas as follow:

In $\mathcal{S}_1$, the equations in $Verify$ are based on the facts as follows:

$$e(z_i^{(1)},H'\cdot H^x)=e(G,H)\cdot e(G_{i,x_i},H_i'\cdot H_i^x)=e(z_i^{(2)},H'\cdot H^x) \tag{A12}$$

By the equation (A12) and in the bilinear mapping with the prime order p, we know that $z_i^{(1)}=z_i^{(2)}$, where $i\in\{1,2,\cdots,n\}$ is uniquely determined in the first n equations, so that $z^{(1)}=z^{(2)}$. and

$$\begin{aligned} y_i^{(1)} &= e(z_i^{(1)},\tilde{H}_i)\cdot e(\tilde{G}_{i,x_i},H_i)^{-1} \\ &= e(z_i^{(1)},H^{g_k(v_i)})\cdot e(G_{i,x_i}^{g_k(v_i)},H^{r_i})^{-1} \\ &= e(z_i^{(1)},H)\cdot e(G_{i,x_i},H^{r_i})^{-1} \\ &= e(z_i^{(1)},H^{\xi+x})^{\frac{1}{\xi+x}}\cdot e(G_{i,x_i},H^{r_i})^{-1} \\ &= e(z_i^{(1)},H'\cdot H^x)^{\frac{1}{\xi+x}}\cdot e(G_{i,x_i},H^{r_i})^{-1} \\ &= [e(G,H)\cdot e(G_{i,x_i},H_i'\cdot H_i^x)]^{\frac{1}{\xi+x}}\cdot e(G_{i,x_i},H^{r_i})^{-1} \\ &= e(z_i^{(2)},H'\cdot H^x)^{\frac{1}{\xi+x}}\cdot e(G_{i,x_i},H^{r_i})^{-1} \\ &= e(z_i^{(2)},H^{\xi+x})^{\frac{1}{\xi+x}}\cdot e(G_{i,x_i},H^{r_i})^{-1} \\ &= e(z_i^{(2)},H)\cdot e(G_{i,x_i},H^{r_i})^{-1} \\ &= e(z_i^{(2)},H^{g_k(v_i)})\cdot e(G_{i,x_i}^{g_k(v_i)},H^{r_i})^{-1} \\ &= e(z_i^{(2)},\tilde{H}_i)\cdot e(\tilde{G}_{i,x_i},H_i)^{-1} \\ &= y_i^{(2)} \end{aligned} \tag{A13}$$

So we can obviously see that

$$y^{(1)} = \prod_{i=1}^{n} y_i^{\gamma_i} = y^{(2)}. \tag{A14}$$

In the above two formulas, we can see that from the fact that actually we have $n+1$ equations in algorithm $Verify$. In addition, by residual n equations $y_i^{(1)} = y_i^{(2)}$, where $i \in \{1, 2, \cdots, n\}$ and $y = \mathbf{F}(sk, x)$ is uniquely determined by the equation in $Verify$. Hence, we have $y^{(1)} = y^{(2)}$ with the same input x. Therefore, when they both have $f(x) = 1$ and $Verify(pk, x, y^{(1)}, z^{(1)} = 1$ and $Verify(pk, x, y^{(2)}, z^{(2)}) = 1$ set up at the same time, we have $y^{(1)} = y^{(2)}$. Then the *uniqueness* property also follows easily.

Above all, we have this equation as follow,

$$Pr[y^{(1)} \neq y^{(2)} | Verify(pk, x, y^{(1)}, z^{(1)} = 1, Verify(pk, x, y^{(2)}, z^{(2)}) = 1] = 0. \tag{A15}$$

Hence, the theorem holds.

Appendix A.3 Pseudorandomness

Finally, we will see the pseudorandomness proof of CVRF-BF as follow.

Theorem 3 (Pseudorandomnes)**.** Suppose the (ℓ, ϵ, t)-DBDHI Assumption holds in a bilinear group $\mathbb{G}(|\mathbb{G}| = p)$. Then the outputs of our CVRF-BF $(Setup, GenFun, Prove, Verify)$ is (l', ϵ', t')-indistinguish-able against chosen input attack (IND-CIA) security, that is a verifiable random function with running time t' and negligible advantage $\epsilon' = \frac{\epsilon}{2^n}$ through l'-time valid queries for $Prove(\cdot)$.

Proof. Now we prove the $Pseudorandomness$ of CVRF-BF and we continue to use the same symbols and letters above. We still suppose that there exists a PPT distinguisher $\mathcal{A}$, which can distinguish the function value $F(sk, x)$ from random choice of group $\mathbb{G}_T$ with the probability $\frac{1}{2} + \epsilon$ and runs in time t. Then we will use the distinguisher $\mathcal{A}$ to construct an PPT algorithm $\mathcal{B}$ (called simulator) with t'-time and advantage $\mathrm{Adv}_{CVRF}^{IND-CIA}(\mathcal{B}) = \frac{\epsilon}{2^n}$ in a very similar way, which can break the ℓ-DBDHI assumption in $\mathbb{G}$.

Next, we give the concrete steps of simulator $\mathcal{B}$ as follow:

- **Input**: The simulator $\mathcal{B}$ takes as input an instance of l-DBDHI problem, as follows:

1. The simulator $\mathcal{B}$ is given the tuple $(\widehat{G}, \widehat{G}^{\eta}, \widehat{G}^{\eta^2}, \cdots, \widehat{G}^{\eta^{\ell}}, \widehat{T_k}) \in \mathbb{G}^{\ell+1} \times \mathbb{G}_T$ as input, where η is an unknown element $\mathbb{Z}_p^*$ and

$$\widehat{T_k} = \begin{cases} e(\widehat{G}, \widehat{G})^{\frac{1}{\eta}}, & k = 0, \\ \widehat{T_1} \leftarrow_R \mathbb{G}_T, & k = 1. \end{cases} \tag{A16}$$

   This means that $\widehat{T_k}$ is either $e(\widehat{G}, \widehat{G})^{\frac{1}{\eta}}$ or a random element in $\mathbb{G}_T$. And in both cases, the probability is equal. Finally, the simulator $\mathcal{B}$ outputs 1 if $\widehat{T} = e(\widehat{G}, \widehat{G})^{\frac{1}{\eta}}$ and 0 otherwise.

- **Setup**: The simulator $\mathcal{B}$ tries to guess the algorithm $\mathcal{A}$'s challenge input by choosing a random message $x^* \in \{0,1\}^n$ and set $\xi = \eta - x^*$. Since η is secret, ξ is secret either. At first, $\mathcal{B}$ produces the output pk of **Setup** as follows:

1. Using binomial theorem and the inputted item $(\widehat{G}, \widehat{G}^{\eta}, \widehat{G}^{\eta^2}, \cdots, \widehat{G}^{\eta^{\ell}})$, the simulator can computes $(\widehat{G}^{\xi}, \widehat{G}^{\xi^2}, \cdots, \widehat{G}^{\xi^{\ell}})$, where $\widehat{G}^{\xi} = \widehat{G}^{\eta - x^*} = \widehat{G}^{\eta} / \widehat{G}^{x^*}$, $\widehat{G}^{\xi^2} = \widehat{G}^{(\eta - x^*)^2} = \widehat{G}^{\eta^2} \widehat{G}^{x^{*2}} / (\widehat{G}^{\eta})^{2x^*}$, and so on.
2. The simulator chooses a set of t random strings $S = \{x_1, \cdots, x_s\}$ from $\{0,1\}^n$ and $x_i \neq x^*$ for all $x_i \in S$. And then it defines polynomial $q(z) = \prod_{w \in S}(z + w) = \sum_{i=0}^{s} a_i z^i \pmod p$, for some $a_i \in \mathbb{Z}_p$. So that, in our CVRF-BF, the simulator defines the generator G from $(\widehat{G}, \widehat{G}^{\xi}, \widehat{G}^{\xi^2}, \cdots, \widehat{G}^{\xi^{\ell}})$ as

$$G = \widehat{G}^{q(\xi)} = \widehat{G}^{\prod_{w \in S}(\xi + w)} = \prod_{i=0}^{s} (\widehat{G}^{\xi^i})^{a_i}.$$

   as well as $H = \widehat{G}^r$, where r is randomly chosen from $\mathbb{Z}_p^*$.
3. The simulator chooses the random $\lambda_i, \psi_i, r_i \in \mathbb{Z}_p$ for $i \in \{1, 2, \cdots, n\}$, and computes the public key

$$pk = \begin{cases} H' = H^{\xi} = (\widehat{G}^{\eta})^r / \widehat{G}^{rx^*}, \\ H_i = H^{r_i} = \widehat{G}^{rr_i}, \\ H_i' = H^{\xi r_i} = (\widehat{G}^{\eta})^{rr_i} / \widehat{G}^{rr_i x^*}, & i \in \{1, 2, \cdots, n\}. \\ G_{i,0} = G^{\lambda_i} = \widehat{G}^{q(\xi)\lambda_i}, \\ G_{i,1} = G^{\psi_i} = \widehat{G}^{q(\xi)\psi_i}. \end{cases}$$

Next, the simulator produces the output pk_f of **GenFun** for a given Boolean function f as follows:

1. The simulator chooses a polynomial $g(z) = \sum_{i=0}^{n} c_i z^i$ with $c_0 = \xi$, $c_i \in_R \mathbb{Z}_p$ for $i = 1, 2, \cdots, n$ and chooses $\lambda_i, \psi_i, v_i \in_R \mathbb{Z}_p$, then computes $G_{1_{i,0}} = G_1^{\lambda_i}$,$G_{1_{i,1}} = G_1^{\lambda_i+\psi_i}$, $G'_{1_{i,b}} = G_{1_{i,b}}^{g(v_i)}$, and $H'_{1_i} = H_1^{g(v_i)}$ for $i = 1, 2, \cdots, n$ and $b = 0, 1$.
2. To do the last step, the simulator need to compute $G_1^{\xi} = G^{\xi f_1(\xi)} = \prod_{i=0}^{\ell-2}(G^{\xi^{i+1}})^{a_i}$ and $H_1^{\xi} = G_1^{\xi^2 q} = (\prod_{i=0}^{\ell-2}(G^{\xi^{i+2}})^{a_i})^q$.
3. Sets the secret key be ξ, and public key be

$$pk_f = \begin{cases} \tilde{H}_i^{(k)} = H^{g_k(v_i)} = (\widehat{G}^{\xi})^r \cdot \widehat{G}^{r\sum_{i=1}^{n} c_i v_i^i} \\ \tilde{G}_{i,x_i}^{(k)} = G_{i,x_i}^{g_k(v_i)} = \begin{cases} \prod_{i=1}^{s+1}(\widehat{G}^{\xi})^{a_i\lambda_i} \cdot (\widehat{G}^{q(\xi)})^{\lambda_i \sum_{i=1}^{n} c_i v_i^i} & x_i = 0, \\ \prod_{i=1}^{s+1}(\widehat{G}^{\xi})^{a_i\psi_i} \cdot (\widehat{G}^{q(\xi)})^{\psi_i \sum_{i=1}^{n} c_i v_i^i} & x_i = 1. \end{cases} \end{cases} \tag{A17}$$

• **Learning 1**: The adversary $\mathcal{A}$ is allowed to ask polynomial number input of VRF queries $\bar{x}$ with the exception that $\bar{x} \neq x^*$. If the queries $\bar{x}$ is not in the set S, the simulator does not make a response for the query $\bar{x} \in S$. Otherwise, the simulator responds the function value and the witness as follows: Let $f'_k(z) = \frac{q(z)}{z+\bar{x_k}} = \prod_{w \in S/\{\bar{x_k}\}}(z+w) = \sum_{i=0}^{s-1} b_i^{(k)} z^i \pmod p$, $k \in \{1, 2, \cdots, s\}$.

$$\begin{cases} \bar{y_k} = \mathbf{F}(sk, \bar{x_k}) = e(\prod_{i=0}^{s-1}(\widehat{G}^{\xi^i})^{b_i^{(k)}}, \widehat{G}^{\xi r}) \\ \bar{z_k} = \mathbf{G}(sk, \bar{x_k}) = (z_1, z_2, \cdots, z_n) \end{cases}, \tag{A18}$$

where

$$z_i = \begin{cases} \widehat{G}^{\frac{q(\xi)}{\xi+\bar{x_k}}} \cdot \widehat{G}^{q(\xi) r_i \lambda_i}, & x_i = 0, \\ \widehat{G}^{\frac{q(\xi)}{\xi+\bar{x_k}}} \cdot \widehat{G}^{q(\xi) r_i \psi_i}, & x_i = 1. \end{cases} \tag{A19}$$

where $\bar{y_k}$ and $\bar{z_k}$ can be effectively calculated.

• **Challenge**: The adversary $\mathcal{A}$ outputs a string x which is not equal to any queries he has asked. If $x \neq x^*$, the simulator halts and outputs the error. Otherwise, this means that x is the same as the *Challenge* x^* of $\mathcal{B}$. The simulator define polynomial $q'(z) = \frac{q(z)z}{z+x^*} - \frac{\delta}{z+x^*} = \sum_{i=0}^{s} \delta_i z^i$ for some $\delta \neq 0$ and $\delta_i \in \mathbb{Z}$. Denote $c = \sum_{j=1}^{t-1} c_j v_i^j$, we compute

$$y^* = \mathbf{F}(sk, x^*) = e(G, H)^{\frac{\xi}{\xi+x^*}} = e(\widehat{G}, \widehat{G})^{r q'(\xi)} \cdot \widehat{T}^{r\delta}, \tag{A20}$$

And if $\widehat{G}_0$ is uniformly distributed, so is y^*. It is the form of our function value.

• **Learning 2** This is the same as Learning 1 with the restriction that the adversary are not allowed to query the challenge input x^*.

• **Guess**: Eventually, $\mathcal{A}$ outputs its guess b'. Then $\mathcal{B}$ outputs the same bit b' as its guess.

$$b' = \begin{cases} 1, & y^* \leftarrow_R W \\ 0, & y^* = F(sk, x^*) \end{cases} \tag{A21}$$

Now we analyze the effectiveness of the algorithm. When the adversary $\mathcal{A}$ runs the Learning process, $\bar{y_k}$ and $\bar{z_k}$ can be effectively calculated and adversary $\mathcal{A}$ will get the correct values.

$$\begin{cases} \bar{y_k} = \mathbf{F}(sk, \bar{x_k}) = e(G, H)^{\frac{\xi}{\xi+\bar{x_k}}} = e(\widehat{G}^{q(\xi)}, \widehat{G}^r)^{\frac{\xi}{\xi+\bar{x_k}}} = e(\prod_{i=0}^{s-1}(\widehat{G}^{\xi^i})^{b_i^{(k)}}, \widehat{G}^{\xi r}), \\ \bar{z_k} = \mathbf{G}(sk, \bar{x_k}) = (z_1, z_2, \cdots, z_n) = G^{\frac{1}{\xi+\bar{x_k}}} \cdot (G_{1,x_1}^{r_1}, G_{2,x_2}^{r_2}, \cdots, G_{n,x_n}^{r_n}), \end{cases} \tag{A22}$$

where

$$z_i = G^{\frac{1}{\xi+\bar{x_k}}} \cdot G_{i,x_i}^{r_i} = \widehat{G}^{\frac{q(\xi)}{\xi+\bar{x_k}}} \cdot G_{i,x_i}^{r_i} = \begin{cases} \widehat{G}^{\frac{q(\xi)}{\xi+\bar{x_k}}} \cdot \widehat{G}^{q(\xi) r_i \lambda_i}, & x_i = 0, \\ \widehat{G}^{\frac{q(\xi)}{\xi+\bar{x_k}}} \cdot \widehat{G}^{q(\xi) r_i \psi_i}, & x_i = 1. \end{cases} \tag{A23}$$

In the Challenge process, y^* and z^* specially as follow,

$$\begin{aligned} y^* &= \mathbf{F}(sk, x^*) = e(G, H)^{\frac{\xi}{\xi+x^*}} = e(\widehat{G}^{q(\xi)}, \widehat{G}^r)^{\frac{\xi}{\xi+x^*}} \\ &= e(\widehat{G}, \widehat{G})^{\frac{q(\xi)\xi r}{\xi+x^*}} = e(\widehat{G}, \widehat{G})^{r(\frac{\delta}{\xi+x^*} + \sum_{i=0}^{s} \delta_i \xi^i)} \\ &= e(\prod_{i=0}^{s}(\widehat{G}^{\xi^i})^{\delta_i}, \widehat{G})^r \cdot e(\widehat{G}, \widehat{G})^{r(\frac{\delta}{\xi+x^*})} \\ &= e(\prod_{i=0}^{s}(\widehat{G}^{\xi^i})^{\delta_i}, \widehat{G})^r \cdot \widehat{T}^{r\delta} = e(\widehat{G}, \widehat{G})^{r q'(\xi)} \cdot \widehat{T}^{r\delta}, \end{aligned} \tag{A24}$$

Furthermore, we have the witness as:

$$z^* = \mathbf{G}(sk, x^*) = (z_1, z_2, \cdots, z_n) = G^{\frac{1}{\xi + x^*}} \cdot (G_{1,x_1}^{r_1}, G_{2,x_2}^{r_2}, \cdots, G_{n,x_n}^{r_n}), \tag{A25}$$

where, $G^{\frac{1}{\xi+x^*}}$ can not be directly calculated, so does z^*. If it could be calculated, in the verify function, $y_i = e(z_i, \tilde{H}_i) \cdot e(\tilde{G}_{i,x_i}, H_i)^{-1}$ will be verified. Unforgeable property of z ensures Pseudorandomness of our CVRF.

Now we analyze the parameters of the algorithm.

From the Game above, $\widehat{T_k}$ is either $e(\widehat{G}, \widehat{G})^{\frac{1}{\eta}}$ or a random element in $\mathbb{G}_T$. And in both cases, the probability is equal. So we have

$$Pr[\widehat{T_0} = e(\widehat{G}, \widehat{G})^{\frac{1}{\eta}}] = Pr[\widehat{T_1} \leftarrow \mathbb{G}_T] = \frac{1}{2}. \tag{A26}$$

Therefore, we have the equation as follow to completes the proof.

$$\begin{aligned}
&\mathrm{Adv}_{CVRF}^{IND-CIA}(\mathcal{B}) + \frac{1}{2} = \Pr\left[b = b' \left| \begin{array}{c} \widehat{G} \leftarrow \mathbb{G}; a \leftarrow \mathbb{Z}_p^*; \\ \widehat{T_0} = e(\widehat{G}, \widehat{G})^{\frac{1}{\eta}}, \widehat{T_1} \leftarrow \mathbb{G}_T; b \leftarrow \{0,1\}; \\ b' \leftarrow \mathcal{B}(\widehat{G}, \widehat{G}^{\eta}, \widehat{G}^{\eta^2}, \cdots, \widehat{G}^{\eta^{\ell}}, \widehat{T_b}); \end{array} \right.\right] \\
&= \Pr\left[b' = 0 \left| \begin{array}{c} \widehat{G} \leftarrow \mathbb{G}; a \leftarrow \mathbb{Z}_p^*; \widehat{T_0} = e(\widehat{G}, \widehat{G})^{\frac{1}{\eta}}; \\ b' \leftarrow \mathcal{B}(\widehat{G}, \widehat{G}^{\eta}, \widehat{G}^{\eta^2}, \cdots, \widehat{G}^{\eta^{\ell}}, \widehat{T_0}); \end{array} \right.\right] \cdot Pr[\widehat{T_0} = e(\widehat{G}, \widehat{G})^{\frac{1}{\eta}}] + \\
&\quad \Pr\left[b' = 1 \left| \begin{array}{c} G \leftarrow \mathbb{G}; a \leftarrow \mathbb{Z}_p^*; \widehat{T_1} \leftarrow \mathbb{G}_T; \\ b' \leftarrow \mathcal{B}(\widehat{G}, \widehat{G}^{\eta}, \widehat{G}^{\eta^2}, \cdots, \widehat{G}^{\eta^{\ell}}, \widehat{T_1}); \end{array} \right.\right] \cdot Pr[\widehat{T_1} \leftarrow \mathbb{G}_T] \\
&= \Pr\left[\mathcal{A}(pk, y^*) = 0 \left| \begin{array}{c} (pk, sk) \leftarrow Setup(1^{\kappa}); \\ x^* \leftarrow \mathcal{A}^{Prove(\cdot)}(pk); \\ y^* \leftarrow F(sk, x^*); \end{array} \right.\right] \cdot Pr[\widehat{T_0} = e(\widehat{G}, \widehat{G})^{\frac{1}{\eta}}] + \\
&\quad \Pr\left[\mathcal{A}(pk, y^*) = 1 \left| \begin{array}{c} (pk, sk) \leftarrow Setup(1^{\kappa}); \\ x^* \leftarrow \mathcal{A}^{Prove(\cdot)}(pk); \\ y^* \leftarrow W; \end{array} \right.\right] \cdot Pr[\widehat{T_1} \leftarrow \mathbb{G}_T] \\
&= \frac{1}{2}\left(\Pr\left[\mathcal{A}(pk, y^*) = 0 \left| \begin{array}{c} (pk, sk) \leftarrow Setup(1^{\kappa}); \\ x^* \leftarrow \mathcal{A}^{Prove(\cdot)}(pk); \\ y^* \leftarrow F(sk, x^*); \end{array} \right.\right] + \Pr\left[\mathcal{A}(pk, y^*) = 1 \left| \begin{array}{c} (pk, sk) \leftarrow Setup(1^{\kappa}); \\ x^* \leftarrow \mathcal{A}^{Prove(\cdot)}(pk); \\ y^* \leftarrow W; \end{array} \right.\right]\right) \\
&= \frac{1}{2} + \frac{1}{2}\left(\Pr\left[\mathcal{A}(pk, y^*) = 0 \left| \begin{array}{c} (pk, sk) \leftarrow Setup(1^{\kappa}); \\ x^* \leftarrow \mathcal{A}^{Prove(\cdot)}(pk); \\ y^* \leftarrow F(sk, x^*); \end{array} \right.\right] - \Pr\left[\mathcal{A}(pk, y^*) = 0 \left| \begin{array}{c} (pk, sk) \leftarrow Setup(1^{\kappa}); \\ x^* \leftarrow \mathcal{A}^{Prove(\cdot)}(pk); \\ y^* \leftarrow W; \end{array} \right.\right]\right).
\end{aligned} \tag{A27}$$

And then, considering the probability of guessing $x = x^*$, that the simulator tries to guess the algorithm $\mathcal{A}$'s challenge input by choosing a random message $x^* \in \{0,1\}^n$, but $x^* \notin S$ and $|\{0,1\}^n \setminus S| = 2^n - t$. So we have

$$Pr[x = x^*] = \frac{1}{2^n - t} \tag{A28}$$

And after $\mathcal{A}$ proceeds the Learning 1, we continue to analyze the above equation as

$$
\begin{aligned}
&= \tfrac{1}{2}+\tfrac{1}{2}\left(\begin{array}{l}\Pr\left[\mathcal{A}(pk,y^*)=0 \left| \begin{array}{c}(pk,sk)\leftarrow Setup(1^\kappa);\\ x^*\leftarrow \mathcal{A}^{Prove(\cdot)}(pk);\\ y^*\leftarrow F(sk,x^*);\end{array}\right.\right]\cdot \Pr[x=x^*]-\\ \Pr\left[\mathcal{A}(pk,y^*)=0 \left| \begin{array}{c}(pk,sk)\leftarrow Setup(1^\kappa);\\ x^*\leftarrow \mathcal{A}^{Prove(\cdot)}(pk);\\ y^*\leftarrow W;\end{array}\right.\right]\cdot \Pr[x=x^*]\end{array}\right)\\
&= \tfrac{1}{2}+\tfrac{1}{2}\cdot\tfrac{1}{2^n-t}\left(\Pr\left[\mathcal{A}(pk,y^*)=0 \left| \begin{array}{c}(pk,sk)\leftarrow Setup(1^\kappa);\\ x^*\leftarrow \mathcal{A}^{Prove(\cdot)}(pk);\\ y^*\leftarrow F(sk,x^*);\end{array}\right.\right]-\Pr\left[\mathcal{A}(pk,y^*)=0 \left| \begin{array}{c}(pk,sk)\leftarrow Setup(1^\kappa);\\ x^*\leftarrow \mathcal{A}^{Prove(\cdot)}(pk);\\ y^*\leftarrow W;\end{array}\right.\right]\right)\\
&\leqslant \tfrac{1}{2}+\tfrac{1}{2^n}\left(\Pr\left[\mathcal{A}(pk,y^*)=0 \left| \begin{array}{c}(pk,sk)\leftarrow Setup(1^\kappa);\\ x^*\leftarrow \mathcal{A}^{Prove(\cdot)}(pk);\\ y^*\leftarrow F(sk,x^*);\end{array}\right.\right]-\Pr\left[\mathcal{A}(pk,y^*)=0 \left| \begin{array}{c}(pk,sk)\leftarrow Setup(1^\kappa);\\ x^*\leftarrow \mathcal{A}^{Prove(\cdot)}(pk);\\ y^*\leftarrow W;\end{array}\right.\right]\right)\\
&= \tfrac{1}{2}+\tfrac{1}{2^n}\left(\Pr\left[b'=0 \left| \begin{array}{c}(pk,sk)\leftarrow Setup(1^\kappa);\\ x^*\leftarrow \mathcal{A}^{Prove(\cdot)}(pk);\\ y_0^*\leftarrow F(sk,x^*);b\leftarrow_R\{0,1\};\\ \mathcal{A}(pk,y_b^*)=b'\end{array}\right.\right]-\Pr\left[b'=0 \left| \begin{array}{c}(pk,sk)\leftarrow Setup(1^\kappa);\\ x^*\leftarrow \mathcal{A}^{Prove(\cdot)}(pk);\\ y_1^*\leftarrow W;b\leftarrow_R\{0,1\};\\ \mathcal{A}(pk,y_b^*)=b'\end{array}\right.\right]\right)\\
&= \tfrac{1}{2}+\tfrac{1}{2^n}\left(\Pr\left[b=b' \left| \begin{array}{c}(pk,sk)\leftarrow Setup(1^\kappa);\\ x^*\leftarrow \mathcal{A}^{Prove(\cdot)}(pk);\\ y_0^*\leftarrow F(sk,x^*);y_1^*\leftarrow W;b\leftarrow\{0,1\};\\ \mathcal{A}(pk,y_b^*)=b'\end{array}\right.\right]-\tfrac{1}{2}\right)\\
&= \tfrac{1}{2}+\tfrac{1}{2^n}\mathrm{Adv}_{CVRF}^{IND-CIA}(\mathcal{A})\leqslant \tfrac{1}{2}+\tfrac{\epsilon}{2^n}.
\end{aligned}
\tag{A29}
$$

Thus, if the advantage of $\mathcal{A}$ is ϵ, the advantage of $\mathcal{B}$ will be $\epsilon/2^n$.

The running time of the reduction is dominated by simulating oracle queries. Per every query, we must perform one bilinear map evaluation (this takes t' time) and $|Q|t_q$ multiplications and exponentiations. Its running time is dominated by answering oracle queries, and each query takes $n\cdot poly(k)$ time to answer. Because $\mathcal{A}$ can make at most t queries. Therefore, $\mathcal{B}$ will run in roughly $t=\mathcal{O}(t'+t_s+t_c+|Q|t_q)$ time. Where t_s is the running time of Setup and t_c is the running time of Challenge, and they are similar to the running time of each query t_q. According to the number of queries $|Q|\leqslant s$, finally we have $t'=\mathcal{O}(t'+st_q)$.

Above all, we have the conclusion that there is no t-time algorithm $\mathcal{A}$ solving the ℓ-DBDHI problem with at least ϵ advantage, and then (ℓ,ϵ,t)-DBDHI assumption holds in $\mathbb{G}$. Hence, the theorem holds.

□

References

1 Silvio Micali, Michael Rabin, and Salil Vadhan. Verifiable random functions. In *40th Annual Symposium on Foundations of Computer Science*, IEEE, 1999:120–130.

2 Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. *Public Key Cryptography*, Springer, 2005: 416–431.

3 Veronika Kuchta and Mark Manulis. Unique Aggregate Signatures with Applications to Distributed Verifiable Random Functions. *International Conference on Cryptology and Network Security*, Springer, 2013: 251–270.

4 Dennis Hofheinz and Tibor Jager. Verifiable Random Functions from Standard Assumptions. *Theory of Cryptography Conference*, Springer, 2016: 336–362.

5 Jorge Antezana, Jordi Marzo, and Jan-Fredrik Olsen. Zeros of random functions generated with de branges kernels. *International Mathematics Research Notices*, 2017(8): 2284-2299.

6 Silvio Micali and Leonid Reyzin. Soundness in the Public-Key Model,CRYPTO,2001,542-565,

7 Joe Kilian, Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings,CRYPTO,2001, 3-540-42456-3.

8 Silvio Micali and Ronald L. Rivest, Micropayments Revisited,CT-RSA,2002, 149-163.

9 Bart Preneel, Topics in Cryptology - CT-RSA 2002, The Cryptographer's Track at the RSA Conference, 2002, San Jose, CA, USA, February 18-22, 2002, Proceedings

10 Stanislaw Jarecki and Vitaly Shmatikov, Handcuffing Big Brother: an Abuse-Resilient Transaction Escrow Scheme, EUROCRYPT, 2004, 590-608.

11 Christian Cachin and Jan Camenisch, Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings, EUROCRYPT, 2004, 3-540-21935-8.

12 Michel Abdalla and Dario Catalano and Dario Fiore, Verifiable Random Functions from Identity-Based Key Encapsulation, EUROCRYPT, 2009, 554-571.

13 Susan Hohenberger and Brent Waters, Constructing Verifiable Random Functions with Large Input Spaces, EUROCRYPT, 2010, 656-672,

14 Henri Gilbert, Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings, 2010, 978-3-642-13189-9c

15 C. Baier and P.D. Argenio and M. Groesser, Partial Order Reduction for Probabilistic Branching Time, Proceedings of the Third Workshop on Quantitative Aspects of Programming Languages, 2006, 97-116

16 M. Groesser and C. Baier, Partial Order Reduction for Markov Decision Processes, Proceedings of the 4th International Symposium on Formal Methods for Components and Objects,2006,408-427

17 D. Shasha and J.T.L. Wang and R. Giugno,Proceedings of the twenty-first ACM Symposium on Principles of Database Systems, Algorithmics and Applications of Tree and Graph Searching, 2002, 39-52

18 Yan Zhu and Gail-Joon Ahn and Hongxin Hu and Huaixi Wang, Cryptographic Role-based Security Mechanisms based on Role-Key Hierarchy, Proceedings of the 2010 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2010, Beijing, China, April 13-16 (ASIACCS)

19 M. Mowbray, Dominator-tree analysis for distributed authorization,Proceedings of the third ACM workshop on Programming,2008,101-112.

20 S. Flesca and S. Greco,Partially Ordered Regular Languages for Graph Queries,Journal of Computer and System Sciences,2005,1-25.

21 M.B. Dwyer and J. Hatcliff and Robby and V.P. Ranganath, Exploiting Object Escape and Locking Information in Partial-Order Reductions for Concurrent Object-Oriented Programs, Formal Methods in System Design,2004,199-240