Functional Signcryption: Notion, Construction, and Applications

by

Pratish Datta

joint work with

Ratna Dutta and **Sourav Mukhopadhyay**

Department of Mathematics
Indian Institute of Technology Kharagpur
Kharagpur-721302
India

ProvSec 2015
24–26th November, 2015

Outline

Motivation

- Functional encryption (FE) enables sophisticated control over decryption rights in multi-user environments.
- Functional signature (FS) allows to enforce complex constraints on signing capabilities.
- *Functional signcryption* (FSC) is a new cryptographic paradigm that aims to provide the functionalities of both FE and FS in an *unified cost-effective primitive*.

The Notion of Functional Signcryption (FSC)

- A trusted authority holds a master secret key and publishes system public parameters.
- Using its master secret key, the authority can provide a signing key $\mathrm{SK}(f)$ for some signing function f to a signcrypter while a decryption key $\mathrm{DK}(g)$ for some decryption function g to a decrypter.
- $\mathrm{SK}(f)$ enables one to signcrypt only messages in the range of f.
- $\mathrm{DK}(g)$ can be utilized to unsigncrypt a ciphertext signcrypting some message m to retrieve $g(m)$ only and to verify the authenticity of the ciphertext at the same time.

A Practical Application of FSC

- Suppose the government is collecting complete photographs of individuals and storing the collected data in a large server for future use by other organization.
- The government is using some photo-processing software that edits the photos and encrypts them before storing to the server.
- It is desirable that the software is allowed to perform only some minor touch-ups of the photos.
- Also, any organization accessing the encrypted database should retrieve only legitimate informations.

A Practical Application of FSC

- The government would provide the photo-processing software the signing keys which allows it to signcrypt original photographs with only the allowable modifications.
- The government would give any organization, wishing to access only informations from the database meeting certain criteria, the corresponding decryption key.
- The decryption key would enable the organization to retrieve only authorized photos and to be convinced that the photos obtained were undergone through only minor photo-editing modifications.

Cryptographic Building Blocks

- $\mathcal{O}$: An indistinguishability obfuscator for P/poly.
- PKE: A CPA-secure public key encryption scheme with message space $\mathbb{M} \subseteq \{0,1\}^{n(\lambda)}$, for some polynomial n.
- SIG: An existentially unforgeable signature scheme with message space $\{0,1\}^{\lambda}$.
- SSS-NIZKPoK: A statistically simulation-sound non-interactive zero-knowledge proof of knowledge system for some NP relation.

Background

Indistinguishability Obfuscation (IO)

An indistinguishability obfuscator (IO) $\mathcal{O}$ for a circuit class $\{\mathbb{C}_\lambda\}$ is a PPT uniform algorithm satisfying the following conditions:

- For any λ, $\mathcal{O}(1^\lambda, C)$ preserves the functionality of the input circuit C, for all $C \in \mathbb{C}_\lambda$.
- For any λ and any two circuits $C_0, C_1 \in \mathbb{C}_\lambda$ with the same functionality, the circuits $\mathcal{O}(1^\lambda, C_0)$ and $\mathcal{O}(1^\lambda, C_1)$ are computationally indistinguishable.

Background

Statistically Simulation-Sound Non-Interactive Zero-Knowledge Proof of Knowledge (SSS-NIZKPoK)

An SSS-NIZKPoK system for $\mathbb{L} \subset \{0,1\}^*$, which is the language containing statements in some binary relation $R \subset \{0,1\}^* \times \{0,1\}^*$, is defined as follows:

- **System Syntax**: SSS-NIZKPoK.Setup, SSS-NIZKPoK.Prove, SSS-NIZKPoK.Verify, SSS-NIZKPoK.SimSetup, SSS-NIZKPoK.SimProve, SSS-NIZKPoK.ExtSetup, SSS-NIZKPoK.Extr.
- **Properties**: perfect completeness, statistical soundness, computational zero-knowledge, knowledge extraction, statistical simulation-soundness.

SSS-NIZKPoK System Used in Our FSC Construction

- We use an SSS-NIZKPoK system for the NP relation R, with statements of the form $X = (\mathrm{PK}^{(1)}_{\mathsf{PKE}}, \mathrm{PK}^{(2)}_{\mathsf{PKE}}, \mathrm{VK}_{\mathsf{SIG}}, e_1, e_2) \in \{0,1\}^*$, witnesses of the form $W = (m, r_1, r_2, f, \sigma, z) \in \{0,1\}^*$, and

$$(X, W) \in R \iff \Big(e_1 = \mathsf{PKE.Encrypt}(\mathrm{PK}^{(1)}_{\mathsf{PKE}}, m; r_1) \bigwedge e_2 = \mathsf{PKE.Encrypt}(\mathrm{PK}^{(2)}_{\mathsf{PKE}}, m; r_2) \bigwedge \mathsf{SIG.Verify}(\mathrm{VK}_{\mathsf{SIG}}, f, \sigma) = 1 \bigwedge m = f(z)\Big),$$

for a function family $\mathbb{F} = \{f : \mathbb{D}_f \to \mathbb{M}\} \subseteq \mathsf{P/poly}$ (with representation in $\{0,1\}^\lambda$).

Construction

FSC.Setup(1^λ)

1. $(\mathrm{PK}^{(1)}_{\mathsf{PKE}}, \mathrm{SK}^{(1)}_{\mathsf{PKE}}), (\mathrm{PK}^{(2)}_{\mathsf{PKE}}, \mathrm{SK}^{(2)}_{\mathsf{PKE}}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$.
2. $(\mathrm{VK}_{\mathsf{SIG}}, \mathrm{SK}_{\mathsf{SIG}}) \leftarrow \mathsf{SIG.KeyGen}(1^\lambda)$.
3. $\mathrm{CRS} \leftarrow \mathsf{SSS\text{-}NIZKPoK.Setup}(1^\lambda)$.
4. Publish $\mathrm{MPK} = (\mathrm{PK}^{(1)}_{\mathsf{PKE}}, \mathrm{PK}^{(2)}_{\mathsf{PKE}}, \mathrm{VK}_{\mathsf{SIG}}, \mathrm{CRS})$.
   Keep $\mathrm{MSK} = (\mathrm{SK}^{(1)}_{\mathsf{PKE}}, \mathrm{SK}_{\mathsf{SIG}})$.

Construction

FSC.SKeyGen$(\textsc{mpk}, \textsc{msk}, f \in \mathbb{F})$

1. $\sigma \leftarrow$ SIG.Sign$(\textsc{sk}_{\mathsf{SIG}}, f)$.
2. Return $\textsc{sk}(f) = (f, \sigma)$ to the legitimate signcrypter.

Construction

FSC.Signcrypt$\big(\text{MPK}, \text{SK}(f) = (f, \sigma), z \in \mathbb{D}_f\big)$

1. $e_\ell = \mathsf{PKE.Encrypt}(\text{PK}^{(\ell)}_{\mathsf{PKE}}, f(z);\ r_\ell)$ for $\ell = 1, 2$, where r_ℓ is the randomness selected for encryption.
2. $\pi \leftarrow \mathsf{SSS\text{-}NIZKPoK.Prove}\big(\text{CRS}, (X, W)\big)$ where $\big(X = (\text{PK}^{(1)}_{\mathsf{PKE}}, \text{PK}^{(2)}_{\mathsf{PKE}}, \text{VK}_{\mathsf{SIG}}, e_1, e_2), W = (f(z), r_1, r_2, f, \sigma, z)\big) \in R$.
3. Output $\text{CT} = (e_1, e_2, \pi)$.

Construction

FSC.DKeyGen$(\text{MPK}, \text{MSK}, g : \mathbb{M} \to \mathbb{R}_g \in \mathsf{P/poly})$

Programs $P^{(g, \text{SK}^{(1)}_{\mathsf{PKE}}, \text{MPK})}$ and $\widetilde{P}^{(g, \text{SK}^{(2)}_{\mathsf{PKE}}, \text{MPK})}$

$P^{(g, \text{SK}^{(1)}_{\mathsf{PKE}}, \text{MPK})}(e_1, e_2, \pi)$

1. $\text{PK}^{(1)}_{\mathsf{PKE}}, \text{PK}^{(2)}_{\mathsf{PKE}}, \text{VK}_{\mathsf{SIG}}, \text{CRS} \leftarrow \text{MPK}$.
2. Set $X = (\text{PK}^{(1)}_{\mathsf{PKE}}, \text{PK}^{(2)}_{\mathsf{PKE}}, \text{VK}_{\mathsf{SIG}}, e_1, e_2)$.
3. If SSS-NIZKPoK.Verify$(\text{CRS}, X, \pi) = 0$, then output $\perp$.
4. Else, output $g\big(\text{PKE.Decrypt}(\text{SK}^{(1)}_{\mathsf{PKE}}, e_1)\big)$.

$\widetilde{P}^{(g, \text{SK}^{(2)}_{\mathsf{PKE}}, \text{MPK})}((e_1, e_2, \pi)$

1. $\text{PK}^{(1)}_{\mathsf{PKE}}, \text{PK}^{(2)}_{\mathsf{PKE}}, \text{VK}_{\mathsf{SIG}}, \text{CRS} \leftarrow \text{MPK}$.
2. Set $X = (\text{PK}^{(1)}_{\mathsf{PKE}}, \text{PK}^{(2)}_{\mathsf{PKE}}, \text{VK}_{\mathsf{SIG}}, e_1, e_2)$.
3. If SSS-NIZKPoK.Verify$(\text{CRS}, X, \pi) = 0$, then output $\perp$.
4. Else, output $g\big(\text{PKE.Decrypt}(\text{SK}^{(2)}_{\mathsf{PKE}}, e_2)\big)$.

- Provide $\text{DK}(g) = (g, \mathcal{O}(P^{(g, \text{SK}^{(1)}_{\mathsf{PKE}}, \text{MPK})}))$ (circuit size $\max\{|P^{(g, \text{SK}^{(1)}_{\mathsf{PKE}}, \text{MPK})}|, |\widetilde{P}^{(g, \text{SK}^{(2)}_{\mathsf{PKE}}, \text{MPK})}|\}$) to the legitimate decrypter.

Construction

FSC.Unsigncrypt$\left(\mathrm{MPK}, \mathrm{DK}(g) = (g, \mathcal{O}(P^{(g, \mathrm{SK}^{(1)}_{\mathsf{PKE}}, \mathrm{MPK})})), \mathrm{CT} = (e_1, e_2, \pi)\right)$

1. Run $\mathcal{O}(P^{(g, \mathrm{SK}^{(1)}_{\mathsf{PKE}}, \mathrm{MPK})})$ with input (e_1, e_2, π).
2. Output the result.

Security

Theorem (*Message Confidentiality of FSC*)

Assuming IO $\mathcal{O}$ *for* P/poly, CPA-*secure public key encryption* PKE, *along with the statistical simulation-soundness and zero-knowledge properties of* SSS-NIZKPoK *system, our* FSC *scheme is selectively message confidential against* CPA.

Theorem (*Ciphertext Unforgeability of FSC*)

Under the assumption that SIG *is existentially unforgeable against* CMA *and* SSS-NIZKPoK *is a proof of knowledge, our* FSC *construction is selectively ciphertext unforgeable against* CMA.

Some Cryptographic Primitives Derived from FSC

- Attribute-based signcryption (ABSC) supporting arbitrary polynomial-size circuits
- SSS-NIZKPoK system for NP relations
- IO for all polynomial-size circuits

ABSC for General Circuits from FSC

ABSC.Setup(1^λ)

1. $(\mathrm{MPK}, \mathrm{MSK}) \leftarrow \mathsf{FSC.Setup}(1^\lambda)$.
2. Publish $\mathrm{MPK}_{\mathsf{ABSC}} = \mathrm{MPK}$. Keep $\mathrm{MSK}_{\mathsf{ABSC}} = \mathrm{MSK}$.

ABSC for General Circuits from FSC

ABSC.SKeyGen($\text{MPK}_{\text{ABSC}} = \text{MPK}, \text{MSK}_{\text{ABSC}} = \text{MSK}, C^{(\text{SIG})} \in$ P/poly)

1. $\text{SK}(f_{C^{(\text{SIG})}}) \leftarrow$ FSC.SKeyGen$(\text{MPK}, \text{MSK}, f_{C^{(\text{SIG})}})$, where $f_{C^{(\text{SIG})}} : \mathbb{D}_f = \{0,1\}^{n=\nu+\mu+\gamma} \rightarrow \mathbb{M} = \{0,1\}^n \cup \{\perp\}$ is defined as

$$f_{C^{(\text{SIG})}}(y \| \overline{y} \| M) = \begin{cases} y\|\overline{y}\|M, & \text{if } C^{(\text{SIG})}(\overline{y}) = 1 \\ \perp, & \text{otherwise} \end{cases}$$

Here, $y \in \{0,1\}^{\nu}$: decryption attribute string

$\overline{y} \in \{0,1\}^{\mu}$: signature attribute string

$M \in \{0,1\}^{\gamma}$: message

2. Provide $\text{SK}_{\text{ABSC}}(C^{(\text{SIG})}) = \text{SK}(f_{C^{(\text{SIG})}})$ to the legitimate signcrypter.

ABSC for General Circuits from FSC

FSC.DKeyGen($\mathrm{MPK}_{\mathsf{ABSC}} = \mathrm{MPK}, \mathrm{MSK}_{\mathsf{ABSC}} = \mathrm{MSK}, C^{(\mathsf{DEC})} \in$ P/poly)

1. $\mathrm{DK}(g_{C^{(\mathsf{DEC})}}) \leftarrow$ FSC.DKeyGen$(\mathrm{MPK}, \mathrm{MSK}, g_{C^{(\mathsf{DEC})}})$, where $g_{C^{(\mathsf{DEC})}} : \mathbb{M} \rightarrow \mathbb{M}$ is defined as

$$g_{C^{(\mathsf{DEC})}}(y\|\overline{y}\|M) = \begin{cases} y\|\overline{y}\|M, & \text{if } C^{(\mathsf{DEC})}(y) = 1 \\ \perp, & \text{otherwise} \end{cases}$$

2. Give $\mathrm{DK}_{\mathsf{ABSC}}(C^{(\mathsf{DEC})}) = \mathrm{DK}(g_{C^{(\mathsf{DEC})}})$ to the legitimate decrypter.

ABSC for General Circuits from FSC

ABSC.Signcrypt$\big(\mathrm{MPK}_{\mathsf{ABSC}} = \mathrm{MPK}, \mathrm{SK}_{\mathsf{ABSC}}(C^{(\mathsf{SIG})}) = \mathrm{SK}(f_{C^{(\mathsf{SIG})}}), y \in \{0,1\}^{\nu}, \overline{y} \in \{0,1\}^{\mu}, M \in \{0,1\}^{\gamma}\big)$

1. $\mathrm{CT} \leftarrow$ FSC.Signcrypt$\big(\mathrm{MPK}, \mathrm{SK}(f_{C^{(\mathsf{SIG})}}), z = y \| \overline{y} \| M\big)$, if $C^{(\mathsf{SIG})}(\overline{y}) = 1$.
2. Output $\mathrm{CT}_{\mathsf{ABSC}}^{(y,\overline{y})} = (y, \overline{y}, \mathrm{CT})$.

ABSC for General Circuits from FSC

ABSC.Unsigncrypt$\big(\mathrm{MPK}_{\mathsf{ABSC}} = \mathrm{MPK}, \mathrm{DK}_{\mathsf{ABSC}}(C^{(\mathsf{DEC})}) = \mathrm{DK}(g_{C^{(\mathsf{DEC})}}), \mathrm{CT}_{\mathsf{ABSC}}^{(y,\overline{y})} = (y, \overline{y}, \mathrm{CT})\big)$

1. Run FSC.Unsigncrypt$(\mathrm{MPK}, \mathrm{DK}(g_{C^{(\mathsf{DEC})}}), \mathrm{CT})$ to obtain $y' \| \overline{y}' \| M'$ or $\perp$.
2. If $y' \| \overline{y}' \| M'$ is obtained and it holds that $y' = y \ \wedge \ \overline{y}' = \overline{y}$, then output M'. Otherwise, output $\perp$.

ABSC for General Circuits from FSC

Security

Theorem (*Message Confidentiality of ABSC*)

If the underlying FSC *scheme is selectively message confidential against* CPA, *then the proposed* ABSC *scheme is also selectively message confidential against* CPA.

Theorem (*Ciphertext Unforgeability of ABSC*)

If the underlying FSC *scheme is selectively ciphertext unforgeable against* CMA, *then the proposed* ABSC *scheme is also selectively ciphertext unforgeable against* CMA.

Overview of IO Construction Using FSC

- From any selectively secure FSC scheme we can obtain a selectively secure FE scheme by including a signing key in the public parameters of FE for the *identity function* on the message space.
- Recently, Ananth et al. [AJS15] has shown how to construct IO for P/poly from selectively secure FE.
- Following these, we can design an IO for P/poly from FSC.

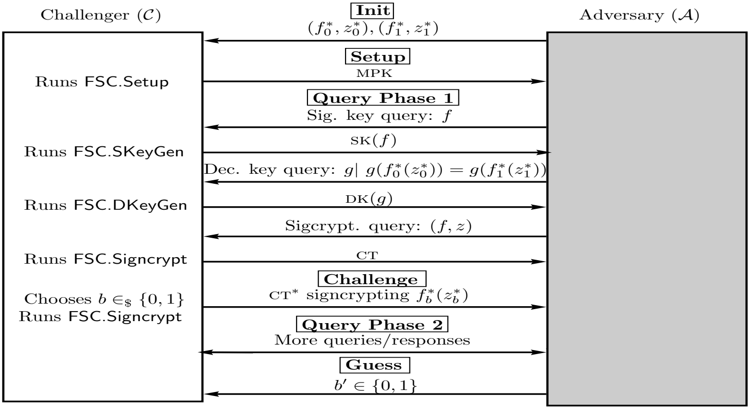
[AJS15]: Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. IACR Cryptology ePrint Archive, 2015.

Future Directions

- Constructing FSC, possibly for restricted classes of functions, from weak and efficient primitives.
- Developing adaptively secure FSC scheme.
- Formulating a simulation-based security notion for FSC.
- Discovering the applications of FSC in building numerous fundamental cryptographic primitives.

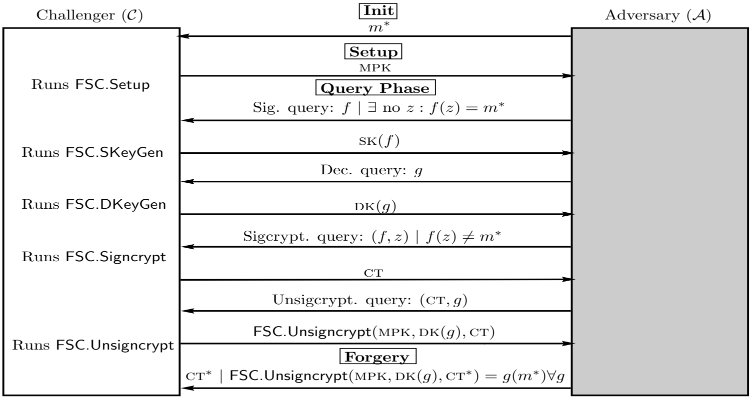
Thanking Note

Selective CPA Message Confidentiality Model for FSC



$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{FSC,s\text{-}IND\text{-}CPA}}(\lambda) = |\Pr[b' = b] - 1/2|$$

Selective CMA Ciphertext Unforgeability Model for FSC



$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{FSC,s\text{-}UF\text{-}CMA}}(\lambda) = \Pr[\mathcal{A} \text{ wins}]$

SSS-NIZKPoK from FSC

SSS-NIZKPoK.Setup(1^λ)

1. $(\text{MPK}, \text{MSK}) \leftarrow$ FSC.Setup(1^λ).
2. Identify some fixed statement $X^* \in \mathbb{L}$.
3. $\text{SK}(f) \leftarrow$ FSC.SKeyGen$(\text{MPK}, \text{MSK}, f)$ and $\text{DK}(g) \leftarrow$ FSC.DKeyGen$(\text{MPK}, \text{MSK}, g)$ respectively for $f : \{0,1\}^{n=\kappa+\rho+1} \rightarrow \mathbb{M} = \{0,1\}^n \cup \{\perp\}$ and $g : \mathbb{M} \rightarrow \{0,1\}^\kappa \cup \{\perp\}$ defined as

$$f(X\|W\|\beta) = \begin{cases} X\|W\|\beta, & \text{if } (X,W) \in R \ \wedge \ \beta = 1 \\ \perp, & \text{otherwise} \end{cases}$$

$$g(X\|W\|\beta) = \begin{cases} X, & \text{if } [(X,W) \in R \ \wedge \ \beta = 1] \ \vee \\ & \quad [X = X^* \ \wedge \ W = 0^\rho \ \wedge \ \beta = 0] \\ \perp, & \text{otherwise} \end{cases}$$

Here $\mathbb{L} \subseteq \{0,1\}^\kappa$ and $\mathbb{R} \subseteq \{0,1\}^\kappa \times \{0,1\}^\rho$.

4. Publish $\text{CRS} = (\text{MPK}, \text{SK}(f), \text{DK}(g))$.

SSS-NIZKPoK from FSC

SSS-NIZKPoK.Prove$\big(\textsc{crs}, (X, W)\big)$

1. $\textsc{ct} \leftarrow$ FSC.Signcrypt$(\textsc{mpk}, \textsc{sk}(f), X \| W \| 1)$.
2. Output $\pi = \textsc{ct}$.

SSS-NIZKPoK from FSC

SSS-NIZKPoK.Verify$(\text{CRS}, X, \pi = \text{CT})$

1. $X' \leftarrow$ FSC.Unsigncrypt$(\text{MPK}, \text{DK}(g), \text{CT})$.
2. Output 1 if $X' = X$. Otherwise, output 0.

SSS-NIZKPoK from FSC

SSS-NIZKPoK.SimSetup$(1^\lambda, \widetilde{X}^*)$

1. $(\text{MPK}, \text{MSK}) \leftarrow$ FSC.Setup(1^λ).
2. $\text{SK}(f) \leftarrow$ FSC.SKeyGen$(\text{MPK}, \text{MSK}, f)$ and $\text{DK}(g) \leftarrow$ FSC.DKeyGen$(\text{MPK}, \text{MSK}, g)$ for functions f and g as in the real setup, where $\widetilde{X}^*$ will play the role of X^*.
3. $\text{SK}(\widetilde{f}) \leftarrow$ FSC.SKeyGen$(\text{MPK}, \text{MSK}, \widetilde{f})$ for $\widetilde{f} : \{0,1\}^n \rightarrow \mathbb{M}$ defined as

$$\widetilde{f}(X\|W\|\beta) = \begin{cases} X\|W\|\beta, & \text{if } [(X, W) \in R \ \wedge \ \beta = 1] \ \vee \\ & \quad [X = \widetilde{X}^* \ \wedge \ W = 0^\rho \ \wedge \ \beta = 0] \\ \perp, & \text{otheriwse} \end{cases}$$

4. Output $\text{CRS} = (\text{MPK}, \text{SK}(f), \text{DK}(g))$ and $\text{TR} = \text{SK}(\widetilde{f})$.

SSS-NIZKPoK from FSC

SSS-NIZKPoK.SimProve$(\textsc{crs}, \textsc{tr}, \widetilde{X}^*)$

1. $\widetilde{\textsc{ct}} \leftarrow$ FSC.Signcrypt$(\textsc{mpk}, \textsc{sk}(\widetilde{f}), \widetilde{X}^* \| 0^\rho \| 0)$.
2. Output $\widetilde{\pi} = \widetilde{\textsc{ct}}$.

SSS-NIZKPoK from FSC

SSS-NIZKPoK.ExtSetup(1^λ)

1. $(\textsc{mpk}, \textsc{msk}) \leftarrow$ FSC.Setup(1^λ).
2. Identify some fixed statement $X^* \in \mathbb{L}$ and compute $\textsc{sk}(f)$ and $\textsc{dk}(g)$ respectively for functions f and g as in the real setup.
3. $\textsc{dk}(g') \leftarrow$ FSC.DKeyGen$(\textsc{mpk}, \textsc{msk}, g')$, where $g' : \{0,1\}^n \rightarrow \{0,1\}^{\rho+1}$ is defined by

$$g'(X\|W\|\beta) = W\|\beta, \text{ for } X\|W\|\beta \in \{0,1\}^n.$$

4. Output $\textsc{crs} = (\textsc{mpk}, \textsc{sk}(f), \textsc{dk}(g))$ and $\widehat{\textsc{tr}} = \textsc{dk}(g')$.

SSS-NIZKPoK from FSC

SSS-NIZKPoK.Extr$(\text{CRS}, \widehat{\text{TR}}, X, \pi = \text{CT})$

1. Run FSC.Unsigncrypt$(\text{MPK}, \text{DK}(g'), \text{CT})$.
2. If $W \| 1 \in \{0,1\}^{\rho+1}$ is obtained, then output W. Otherwise, output $\perp$ indicating failure.

SSS-NIZKPoK from FSC

Security

Theorem

Assuming that the underlying FSC *scheme is selective message confidential against* CPA *and selective ciphertext unforgeable against* CMA, *the described* SSS-NIZKPoK *system satisfies all the criteria of* SSS-NIZKPoK.