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

1 Introduction
2 Our FSC Scheme
3 Cryptographic Primitives from FSC
4 Conclusion
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.
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 $SK(f)$ for some signing function $f$ to a signcrypter while a decryption key $DK(g)$ for some decryption function $g$ to a decrypter.

$SK(f)$ enables one to signcrypt only messages in the range of $f$.

$DK(g)$ 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.
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 \( M \subseteq \{0, 1\}^{n(\lambda)} \), 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.
An indistinguishability obfuscator (IO) $\mathcal{O}$ for a circuit class $\{C_\lambda\}$ is a PPT uniform algorithm satisfying the following conditions:

- For any $\lambda$, $\mathcal{O}(1^\lambda, C)$ preserves the functionality of the input circuit $C$, for all $C \in \mathbb{C}_\lambda$.

- For any $\lambda$ 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:


- **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 \( X = (\text{PK}_{\text{PKE}}^{(1)}, \text{PK}_{\text{PKE}}^{(2)}, \text{VK}_{\text{SIG}}, e_1, e_2) \in \{0, 1\}^* \), witnesses of the form \( W = (m, r_1, r_2, f, \sigma, z) \in \{0, 1\}^* \), and

\[
(X, W) \in R \iff \left( e_1 = \text{PKE.Encrypt}(\text{PK}_{\text{PKE}}^{(1)}, m; r_1) \wedge e_2 = \text{PKE.Encrypt}(\text{PK}_{\text{PKE}}^{(2)}, m; r_2) \wedge \text{SIG.Verify}(\text{VK}_{\text{SIG}}, f, \sigma) = 1 \wedge m = f(z) \right),
\]

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

**FSC.Setup($1^\lambda$)**

1. \((PK_{PKE}^{(1)}, SK_{PKE}^{(1)}), (PK_{PKE}^{(2)}, SK_{PKE}^{(2)}) \leftarrow PKE.KeyGen(1^\lambda)\).

2. \((VK_{SIG}, SK_{SIG}) \leftarrow SIG.KeyGen(1^\lambda)\).

3. \(CRS \leftarrow SSS-NIZKPoK.Setup(1^\lambda)\).

4. **Publish** \(MPK = (PK_{PKE}^{(1)}, PK_{PKE}^{(2)}, VK_{SIG}, CRS)\).
   **Keep** \(MSK = (SK_{PKE}^{(1)}, SK_{SIG})\).
Construction
FSC.SKeyGen\((\text{MPK, MSK, } f \in \mathbb{F})\)

1. \(\sigma \leftarrow \text{SIG.Sign}(SK_{\text{SIG}}, f)\).

2. Return \(SK(f) = (f, \sigma)\) to the legitimate signcrupter.
Construction

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

1. \(e_\ell = \text{PKE.Encrypt}(\text{PK}_{\text{PKE}}^{(\ell)}, f(z); r_\ell)\) for \(\ell = 1, 2\), where \(r_\ell\) is the randomness selected for encryption.

2. \(\pi \leftarrow \text{SSS-NIZKPoK.Prove}((\text{CRS}, (X, W))\) where \((X = (\text{PK}_{\text{PKE}}^{(1)}, \text{PK}_{\text{PKE}}^{(2)},\text{VKS}_{\text{SIG}}, e_1, e_2), W = (f(z), r_1, r_2, f, \sigma, z)) \in R.\)

3. **Output** \(\text{CT} = (e_1, e_2, \pi)\).
Construction

**FSC.DKeyGen**\((\text{MPK}, \text{MSK}, g : \mathbb{M} \rightarrow \mathbb{R}, g \in \text{P/poly})\)

---

**Programs** \(P(g, S_{\text{SK}_{\text{PKE}}}^{(1)}, \text{MPK})\) and \(\tilde{P}(g, S_{\text{SK}_{\text{PKE}}}^{(2)}, \text{MPK})\)

1. \(\text{PK}_{\text{PKE}}^{(1)}, \text{PK}_{\text{PKE}}^{(2)}, \text{VKSIG}, \text{CRS} \leftarrow \text{MPK}.\)
2. Set \(X = (\text{PK}_{\text{PKE}}^{(1)}, \text{PK}_{\text{PKE}}^{(2)}, \text{VKSIG}, e_1, e_2).\)
3. If \(\text{SSS-NIZKPoK.Verify(CRS, X, \pi)} = 0,\) then output \(\bot.\)
4. Else, output \(g\left(\text{PKE}.\text{Decrypt}(S_{\text{SK}_{\text{PKE}}}^{(1)}, e_1)\right).\)

---

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

\[ \text{FSC.Unsigncrypt}(\text{mpk, dk}(g) = (g, O(P(g, SK_{\text{PKE}}, \text{mpk}))), CT = (e_1, e_2, \pi)) \]

1. Run \( O(P(g, SK_{\text{PKE}}, \text{mpk})) \) with input \((e_1, e_2, \pi)\).

2. Output the result.
Theorem (Message Confidentiality of FSC)

Assuming IO 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**($1^\lambda$)

1. \((\text{MPK}, \text{MSK}) \leftarrow \text{FSC.Setup}(1^\lambda)\).

2. **Publish** \(\text{MPK}_{\text{ABSC}} = \text{MPK}\). **Keep** \(\text{MSK}_{\text{ABSC}} = \text{MSK}\).
### ABSC for General Circuits from FSC

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

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

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

Here, \(y \in \{0, 1\}^\nu : \text{decryption attribute string}\)

\(\overline{y} \in \{0, 1\}^{\mu} : \text{signature attribute string}\)

\(M \in \{0, 1\}^{\gamma} : \text{message}\)

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(\(\text{MPK}_{\text{ABSC}} = \text{MPK}, \text{MSK}_{\text{ABSC}} = \text{MSK}, C^{(\text{DEC})} \in \text{P/poly}\))

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

\[
g_{C^{(\text{DEC})}}(y \parallel \overline{y} \parallel M) = \begin{cases} 
  y \parallel \overline{y} \parallel M, & \text{if } C^{(\text{DEC})}(y) = 1 \\
  \bot, & \text{otherwise}
\end{cases}
\]

2. Give \(\text{DK}_{\text{ABSC}}(C^{(\text{DEC})}) = \text{DK}(g_{C^{(\text{DEC})}})\) to the legitimate decrypter.
ABSC for General Circuits from FSC

\[
\text{ABSC.Signcrypt}(\text{MPK}_{\text{ABSC}} = \text{MPK}, \text{SK}_{\text{ABSC}}(C^{(\text{SIG})}) = \text{SK}(f_{C^{(\text{SIG})}}), y \in \{0, 1\}^\nu, \overline{y} \in \{0, 1\}^\mu, M \in \{0, 1\}^\gamma)
\]

1. \( \text{CT} \leftarrow \text{FSC.Signcrypt}(\text{MPK}, \text{SK}(f_{C^{(\text{SIG})}}), z = y \| \overline{y} \| M), \text{ if } C^{(\text{SIG})}(\overline{y}) = 1. \)

2. Output \( \text{CT}_{\text{ABSC}} = (y, \overline{y}, \text{CT}). \)
ABSC for General Circuits from FSC

ABSC-Unsigncrypt(\text{MPK}_{ABSC} = MPK, \text{DK}_{ABSC}(C^{(DEC)})) = \text{DK}(g_{C^{(DEC)}}), CT_{ABSC}^{(y, \overline{y})} = (y, \overline{y}, CT))

1. Run FSC.Unsigncrypt(\text{MPK}, \text{DK}(g_{C^{(DEC)}}), CT) to obtain $y' || \overline{y}' || M'$ or $\perp$.

2. If $y' || \overline{y}' || M'$ is obtained and it holds that $y' = y$ \land \overline{y}' = \overline{y}$, then output $M'$. Otherwise, output $\perp$. 
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.

---

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

Challenger (C)
- Runs FSC.Setup
- Runs FSC.SKeyGen
  - Chooses $b \in \{0, 1\}$
- Runs FSC.Signcrypt

Adversary (A)
- Runs FSC.Setup
- Query Phase 1
  - Sig. key query: $f$
    - $\text{sk}(f)$
  - Dec. key query: $g$ \mid $g(f_0^*(z_0^*)) = g(f_1^*(z_1^*))$
    - $\text{dk}(g)$
- Challenge
  - $\text{ct}^*$ signcrypting $f_b^*(z_b^*)$
- Query Phase 2
  - More queries/responses
- Guess
  - $b' \in \{0, 1\}$

$\text{Adv}^{\text{FSC,s-IND-CPA}}_A(\lambda) = |\Pr[b' = b] - 1/2|$
Selective CMA Ciphertext Unforgeability Model for FSC

Challenger (C)  Adversary (A)

Runs FSC.Setup

Runs FSC.SKeyGen

Runs FSC.DKeyGen

Runs FSC.Signcrypt

Runs FSC.Unsigncrypt

**Init**

m*

**Setup**

MPK

**Query Phase**

Sig. query: \( f \mid \exists \) no \( z : f(z) = m^* \)

Dec. query: \( g \)

**FSC.**Unsigncrypt

\( CT \)

Sigcrypt. query: \( (f, z) | f(z) \neq m^* \)

Unsigcrypt. query: \( (CT, g) \)

\( CT^* | FSC.\)Unsigncrypt(\( MPK, DK(g), CT^* \)) = \( g(m^*) \forall g \)

**Forgery**

Adv_{FSC,s-UF-CMA}(\( \lambda \)) = Pr[A wins]

Pratish Datta

Functional Signcryption

ProvSec 2015 26
SSS-NIZKPoK from FSC

SSS-NIZKPoK.Setup(1^λ)

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

\[
\begin{align*}
    f(X\|W\|\beta) &= \begin{cases} 
    X\|W\|\beta, & \text{if } (X, W) \in R \land \beta = 1 \\
    \bot, & \text{otherwise}
    \end{cases} \\
    g(X\|W\|\beta) &= \begin{cases} 
    X, & \text{if } [(X, W) \in R \land \beta = 1] \lor \\
    [X = X^* \land W = 0^\rho \land \beta = 0] \\
    \bot, & \text{otherwise}
    \end{cases}
\end{align*}
\]

Here \(\mathcal{L} \subseteq \{0, 1\}^\kappa\) and \(\mathcal{R} \subseteq \{0, 1\}^\kappa \times \{0, 1\}^\rho\).
4. Publish \(\text{CRS} = (\text{MPK}, \text{SK}(f), \text{DK}(g)).\)
SSS-NIZKPoK from FSC

SSS-NIZKPoK.Prove\( (\text{crs}, (X, W)) \)

1. \( CT \leftarrow \text{FSC.Signcrypt}(\text{mpk}, \text{sk}(f), X \parallel W \parallel 1) \).

2. Output \( \pi = CT \).
SSS-NIZKPoK from FSC

SSS-NIZKPoK.Verify\((\text{crs}, X, \pi = \text{ct})\)

1. \(X' \leftarrow \text{FSC.Unsigncrypt}(\text{mpk}, \text{DK}(g), \text{ct}).\)

2. Output 1 if \(X' = X\). Otherwise, output 0.
**SSS-NIZKPoK from FSC**

**SSS-NIZKPoK.SimSetup\((1^λ, \tilde{X}^*)\)**

1. \((\text{MPK}, \text{MSK}) \leftarrow \text{FSC.Setup}(1^λ)\).

2. \(\text{SK}(f) \leftarrow \text{FSC.SKeyGen}(\text{MPK}, \text{MSK}, f)\) and \(\text{DK}(g) \leftarrow \text{FSC.DKeyGen}(\text{MPK}, \text{MSK}, g)\) for functions \(f\) and \(g\) as in the real setup, where \(\tilde{X}^*\) will play the role of \(X^*\).

3. \(\text{SK}(\tilde{f}) \leftarrow \text{FSC.SKeyGen}(\text{MPK}, \text{MSK}, \tilde{f})\) for \(\tilde{f}: \{0, 1\}^n \rightarrow Μ\) defined as

\[
\tilde{f}(X\|W\|\beta) = \begin{cases} 
X\|W\|\beta, & \text{if } [(X, W) \in R \land \beta = 1] \lor \\
[X = \tilde{X}^* \land W = 0^ρ \land \beta = 0] & \text{otherwise}
\end{cases}
\]

4. **Output** \(\text{CRS} = (\text{MPK}, \text{SK}(f), \text{DK}(g))\) and \(\text{TR} = \text{SK}(\tilde{f})\).
SSS-NIZKPoK from FSC
SSS-NIZKPoK.SimProve(\texttt{CRS}, \texttt{TR}, \tilde{X}^*)

1. \(\widetilde{CT} \leftarrow \text{FSC.Signcrypt} (\texttt{MPK}, \texttt{SK}(\tilde{f}), \tilde{X}^* \parallel 0^p \parallel 0)\).

2. Output \(\widetilde{\pi} = \widetilde{CT}\).
SSS-NIZKPoK from FSC
SSS-NIZKPoK. ExtSetup(1^λ)

1. \((\text{MPK}, \text{MSK}) \leftarrow \text{FSC.Setup}(1^\lambda)\).

2. Identify some fixed statement \(X^* \in \mathbb{L}\) and compute \(\text{SK}(f)\) and \(\text{DK}(g)\) respectively for functions \(f\) and \(g\) as in the real setup.

3. \(\text{DK}(g') \leftarrow \text{FSC.DKeyGen}(\text{MPK}, \text{MSK}, g')\), 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 \(\text{CRS} = (\text{MPK}, \text{SK}(f), \text{DK}(g))\) and \(\widehat{\text{TR}} = \text{DK}(g').\)
SSS-NIZKPoK from FSC

SSS-NIZKPoK.Extr\((\text{crs}, \tr, X, \pi = \text{ct})\)

1. Run FSC.Unsigncrypt\((\text{mpk}, \text{dk}(g'), \text{ct})\).

2. If \(W \parallel 1 \in \{0, 1\}^{\rho + 1}\) is obtained, then output \(W\). Otherwise, output \(\bot\) indicating failure.
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.