SCIENCE CHINA
Information Sciences

• LETTER •

January 2017, Vol. 60 019101:1–019101:3
doi: 10.1007/s11432-015-0994-0

Attribute-based non-interactive key exchange

Fei TANG[1*], Rui ZHANG[2*] & Hongda LI[2*]

[1]*College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing* 400065*, China;*
[2]*State Key Laboratory of Information Security, Institute of Information Engineering of Chinese Academy of Sciences, Beijing* 100093*, China*

Received December 9, 2015; accepted February 3, 2016; published online May 16, 2016

Citation Tang F, Zhang R, Li H D. Attribute-based non-interactive key exchange. Sci China Inf Sci, 2017, 60(1): 019101, doi: 10.1007/s11432-015-0994-0

Dear editor,

In this work, we study the notion of attribute-based non-interactive key-exchange (ABNIKE). As a natural extension of the notions of non-interactive key-exchange (NIKE) [1, 2] and identity-based non-interactive key-exchange (IB-NIKE) [3, 4], ABNIKE allows users to non-interactively agree on a common shared key. Learning from attribute-based encryption [5], we divide ABNIKE into two forms: key-policy ABNIKE (KP-ABNIKE) and shared-key-policy ABNIKE (SP-ABNIKE). Intuitively, in a KP-ABNIKE scheme, a user who is associated with a policy function f has a secret key sk_f. The shared key K_x will be established according to an attribute set x. A user can non-interactively generate K_x if and only if x satisfies f, i.e., $f(x) = 1$. On the contrary, in an SP-ABNIKE scheme, a user's secret key is associated with an attribute set x, while a shared key is associated with a policy function f. In this work, we first give a formal definition of ABNIKE. We then define the security model for ABNIKE in the dishonest key registration (DKR) setting. Next, by using differing-input obfuscation (diO) [6,7], we construct an ABNIKE scheme. Finally, we show that the notion of ABNIKE implies IBNIKE and two- or more-party ABNIKE, which have been realized in previous work.

Preliminaries. We now present some definitions that will be used for our construction.

Differing-input obfuscation. The definition of diO with auxiliary input follows that of Ananth et al. [6], which is equivalent to that given by Boyle et al. [7]. First, we define the notion of differing-input circuits family.

Definition 1. A circuit family $\mathbb{C}$ with a sampler $(C_0, C_1, \mathrm{aux}) \leftarrow \mathrm{Samp}(1^\lambda)$ that samples $C_0, C_1 \in \mathbb{C}$ is said to be a differing-input family, if for all PPT adversaries $\mathcal{A}$, we have

$$\Pr[C_0(x) \neq C_1(x) : (C_0, C_1, \mathrm{aux}) \leftarrow \mathrm{Samp}(1^\lambda), x \leftarrow \mathcal{A}(1^\lambda, C_0, C_1, \mathrm{aux})] = \mathrm{negl}(\lambda).$$

We now define the notion of diO for a differing-input circuits family.

Definition 2. A uniform PPT machine diO is called a differing-input obfuscator for a differing-input circuits family $\mathbb{C} = \{\mathbb{C}_\lambda\}$ if it satisfies the following properties:

- Correctness. For all security parameters $\lambda \in \mathbb{N}$, all $C \in \mathbb{C}_\lambda$, and all inputs x, we have $\Pr[C'(x) = C(x) : C' \leftarrow \mathrm{diO}(\lambda, C)] = 1$.
- Polynomial slowdown. There exists a universal polynomial poly such that for any circuit $C \in \mathbb{C}_\lambda$ we have $|C'| \leqslant \mathrm{poly}(|C|)$, where $C' = \mathrm{diO}(C)$.

* Corresponding author (email: tangfei127@163.com, r-zhang@iie.ac.cn, lihongda@iie.ac.cn)
The authors declare that they have no conflict of interest.

• Differing-inputs. For any (not necessarily uniform) PPT distinguisher D, all security parameters $\lambda \in \mathbb{N}$, and $(C_0, C_1, \mathrm{aux}) \leftarrow \mathrm{Samp}(1^\lambda)$, we have

$$|\Pr[D(\mathrm{diO}(\lambda, C_0), \mathrm{aux}) = 1] \\ -\Pr[D(\mathrm{diO}(\lambda, C_1), \mathrm{aux}) = 1]| = \mathrm{negl}(\lambda).$$

Punctured pseudorandom functions. In punctured PRF [8], one can derive a punctured key K_S with respect to a subset $S \subseteq \mathbb{D}$ from the secret key K. This punctured key enables the evaluation of the PRF in the subset $\mathbb{D} \backslash S$ of the domain and nowhere else.

Definition 3. A punctured PRF consists of a triple of algorithms, $\mathfrak{F} = (\mathrm{PRF.Key}, \mathrm{PRF.Pun}, F)$, and a pair of computable functions, $n(\cdot)$ and $m(\cdot)$, satisfying the following conditions:

• Functionality preserved under puncturing: for any PPT adversary $\mathcal{A}$ that outputs a set $S \subseteq \mathbb{D}$ where $\mathbb{D} = \{0,1\}^{n(\lambda)}$ is the domain of the punctured PRF, if $x \in \mathbb{D} \backslash S$ then

$$\Pr[F(K, x) = F(K_S, x) : K \leftarrow \mathrm{PRF.Key}(1^\lambda), \\ K_S \leftarrow \mathrm{PRF.Pun}(K, S)] = 1.$$

• Pseudorandom at punctured points: for any PPT adversary $(\mathcal{A}_1, \mathcal{A}_2)$ such that $\mathcal{A}_1(1^\lambda)$ outputs a set $S \subseteq \mathbb{D}$ and a state τ, consider an experiment where $K \leftarrow \mathrm{PRF.Key}(1^\lambda)$ and $K_S \leftarrow \mathrm{PRF.Pun}(K, S)$:

$$\Pr[\mathcal{A}_2(\tau, K_S, S, F(K, S)) = 1] \\ -\Pr[\mathcal{A}_2(\tau, K_S, S, U_{m(\lambda)\cdot|S|}) = 1] = \mathrm{negl}(\lambda),$$

where $S = \{x_1, \ldots, x_k\}$, $F(K, S)$ is the concatenation of the elements of S in lexicographic order, i.e., $F(K, x_1)||\cdots||F(K, x_k)$, and U_ℓ denotes the uniform distribution over ℓ bits.

Digital signatures. A signature scheme $\mathfrak{S}$ consists of the following three PPT algorithms: Sig.Key takes as input a security parameter λ and outputs a signing-verification key pair $(\mathrm{sk}, \mathrm{vk})$; Sig.Sign takes as inputs the signing key sk and a message m, and outputs a signature σ; Sig.Vrfy takes as inputs the verification key vk and a purported signature σ on a message m, and outputs 1 if it is valid or 0 otherwise.

For correctness, it is required that for any $(\mathrm{sk}, \mathrm{vk}) \leftarrow \mathrm{Sig.Key}(1^\lambda)$ and for any message m, $\Pr[\mathrm{Sig.Vrfy}(\mathrm{vk}, m, \mathrm{Sig.Sign}(\mathrm{sk}, m)) = 1] = 1$.

Definition 4. We say that a signature scheme $\mathfrak{S} = (\mathrm{Sig.Key}, \mathrm{Sig.Sign}, \mathrm{Sig.Vrfy})$ is EU-CMA secure [9] if, for any PPT adversary $\mathcal{A}$ with oracle access to Sig.Sign, the probability that, on input of a uniformly chosen verification key vk, $\mathcal{A}$ outputs a pair (m^*, σ^*) such that $\mathrm{Sig.Vrfy}(\mathrm{vk}, m^*, \sigma^*) = 1$ where m^* was not queried to Sig.Sign oracle, is negligible, where the probability is over vk and the randomness of the Sig.Sign oracle.

Definitions. Let $\mathbb{A}$ be the universe of possible attributes. A claimed policy over $\mathbb{A}$ is a Boolean function $f \in \mathbb{F}$, where $\mathbb{F}$ is the space of all possible policy functions. We say that an attribute set $x \subseteq \mathbb{A}$ satisfies a policy function f if $f(x) = 1$.

Inspired by the classification of attribute-based encryption [5], we divide ABNIKE into two flavors, KP-ABNIKE and SP-ABNIKE. For simplicity, we give the definition of KP-ABNIKE. The notion of SP-ABNIKE can be easily obtained by interchanging the function f and attribute set x.

• $\mathrm{AB.Setup}(1^\lambda)$: The setup algorithm takes as input the security parameter λ. It outputs the public parameters pp and the master key msk.

• $\mathrm{AB.KeyGen}(\mathrm{msk}, f)$: The key generation algorithm takes as input the master key msk and a policy function f. It outputs a secret key sk_f.

• $\mathrm{AB.SharedKey}(\mathrm{pp}, \mathrm{sk}_f, f, x)$: Each user can non-interactively generate a common shared key $K_x \in \mathbb{SHK}$ only if $f(x) = 1$, with respect to an attribute set x using the public parameters pp, his secret key sk_f and function f, where $\mathbb{SHK}$ is the share key space.

For correctness, it is required that for all λ, f_0, f_1, x, and all $(\mathrm{pp}, \mathrm{msk}) \leftarrow \mathrm{AB.Setup}(1^\lambda)$, $\mathrm{sk}_{f_0} \leftarrow \mathrm{AB.KeyGen}(\mathrm{msk}, f_0)$, and $\mathrm{sk}_{f_1} \leftarrow \mathrm{AB.KeyGen}(\mathrm{msk}, f_1)$, if $f_0(x) = f_1(x) = 1$, then we have

$$\mathrm{AB.SharedKey}(\mathrm{pp}, \mathrm{sk}_{f_0}, f_0, x) = \\ \mathrm{AB.SharedKey}(\mathrm{pp}, \mathrm{sk}_{f_1}, f_1, x) = K_x.$$

Construction. We now construct an ABNIKE scheme. Our ABNIKE scheme follows the punctured program technique (with diO instead of iO) devised by Sahai and Waters [8]. To generate the secret key for f, we choose a signature scheme $\mathfrak{S} = (\mathrm{Sig.Key}, \mathrm{Sig.Sign}, \mathrm{Sig.Vrfy})$ to sign on f and set the resulting signature as the secret key. In addition, we also choose a punctured PRF $\mathfrak{F} = (\mathrm{PRF.Key}, \mathrm{PRF.Pun}, F)$. For simplicity, we assume that the Sig.Sign and F algorithms will take inputs with appropriate length.

The following is a KP-ABNIKE scheme. By using universal circuits, we can easily construct an SP-ABNIKE scheme.

• $\mathrm{AB.Setup}(1^\lambda)$: The setup algorithm takes as input a security parameter λ and does the following. It first runs the $\mathrm{Sig.Key}(1^\lambda)$ and $\mathrm{PRF.Key}(1^\lambda)$ algorithms to produce a signing-verification key pair $(\mathrm{sk}, \mathrm{vk})$ and a PRF key K, respectively. It then builds an obfuscated program diO($\mathcal{P}$), where the program $\mathcal{P}$ contains two constant values, vk and K. Then, any user can run this program on inputs an attribute set x, his policy function f and secret key sk_f. Formally, the program $\mathcal{P}$ is defined below.

(1) Given inputs (x, f, sk_f), the program first checks that $f(x) \stackrel{?}{=} 1$ and $\mathrm{Sig.Vrfy}(\mathrm{vk}, f, \mathrm{sk}_f) \stackrel{?}{=} 1$ holds or not.

(2) If any checks fail, then it outputs $\perp$; else, it outputs $K_x \leftarrow F(K, x)$.

The public parameters pp consist of the descriptions of the attribute universe $\mathbb{A}$, the space of policy functions $\mathbb{F}$, the shared key space $\mathbb{SHK}$, and the obfuscated program diO($\mathcal{P}$). The master key is $\mathrm{msk} = \mathrm{sk}$.

- $\mathrm{AB.KeyGen}(\mathrm{msk}, f)$: To generate the secret key for a function $f \in \mathbb{F}$, the key generation algorithm runs the signing algorithm of the signature scheme $\mathrm{sk}_f \leftarrow \mathrm{Sig.Sign}(\mathrm{sk}, f)$ and outputs sk_f.
- $\mathrm{AB.SharedKey}(\mathrm{pp}, \mathrm{sk}_f, f, x)$: Each user runs the obfuscated program diO($\mathcal{P}$) on the inputs (x, f, sk_f) and outputs the result.

Conclusion. In this work, we define the notion of attribute-based non-interactive key-exchange. In addition, by using differing-input obfuscation, we give a concrete construction of such cryptographic primitive. Due to space limitations, the security of our construction and further analysis are available in the supporting information.

Acknowledgements The work of Rui Zhang was supported by Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDA06010703). The work of Hongda Li was supported by National Natural Science Foundation of China (Grant No. 60970139).

Supporting information The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Diffie W, Hellman M E. New directions in cryptography. IEEE Trans Inf Theory, 1976, 22: 644–654
2 Freire E S V, Hofheinz D, Kiltz E, et al. Non-interactive key exchange. In: Public-Key Cryptography — PKC 2013. Berlin: Springer, 2013. 254–271
3 Boneh D, Zhandry M. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In: Advances in Cryptology — CRYPTO 2014. Berlin: Springer, 2014. 480–499
4 Sakai R, Ohgishi K, Kasahara M. Cryptosystems based on pairing. In: Proceedings of the Symposium on Cryptography and Information Security, Okinawa, 2000. 135–148
5 Goyal V, Pandey O, Sahai A, et al. Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS 2006), Alexandria, 2006. 89–98
6 Ananth P, Boneh D, Garg S, et al. Deffering-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689. http://eprint.iacr.org/. 2013
7 Boyle E, Chung K M, Pass R. On extractability obfuscation. In: Theory of Cryptography. Berlin: Springer, 2014. 52–73
8 Sahai A, Waters B. How to use indistinguishability obfuscation: deniable encryption, and more. In: Proceedings of the 46th Annual Symposium on Theory of Computing (STOC 2014). New York: ACM, 2014. 475–484
9 Goldwasser S, Micali S, Rivest R L. A digital signature scheme secure against adaptive chosen-message attacks. SIAM J Comput, 1988, 17: 281–308