1-1-2006

Trust negotiation with trust parameters

Fuchun Guo
Fujian Normal University, fuchun@uow.edu.au

Zhide Chen
Fujian Normal University

Yi Mu
University of Wollongong, ymu@uow.edu.au

Trust negotiation with trust parameters

Abstract

The notion of hidden credentials can be applied to protection of sensitive credentials, resources and policies in trust negotiation. It allows the server to encrypt a resource so that only the client with the correct credentials can decrypt it. The existing scheme of hidden credentials requires that the server grant access to the encrypted resource directly to the client during the negotiation without knowing whether or not the client can decrypt it. It would be a burden if the resources were very large. We found that when the server grants access to services rather than resources, the existing hidden credentials schemes are insecure under our policy attacks, since the server can illegally learn the client's credentials from the attack. In this paper, we propose a scheme to stop the server from mounting a policy attack.

Keywords

Trust, negotiation, trust, parameters

Publication Details

Guo, F., Chen, Z. & Mu, Y. (2006). Trust negotiation with trust parameters. In Y. Cheung, Y. Wang & H. Liu (Eds.), International Conference on Computational Intelligence and Security (pp. 1291-1294). Hong Kong: IEEE.

Trust Negotiation with Trust Parameters*

Fuchun Guo
School of Mathematics and Computer Science
Fujian Normal University, Fuzhou, China
fuchunguo1982@163.com

Zhide Chen
School of Mathematics and Computer Science
Fujian Normal University, Fuzhou, China
zdchen@fjnu.edu.cn

Yi Mu
School of Information Technology and Computer Science
University of Wollongong, Wollongong, NSW 2522, Australia
ymu@uow.edu.au

Abstract

The notion of Hidden Credentials can be applied to protection of sensitive credentials, resources and policies in Trust Negotiation. It allows the server to encrypt a resource so that only the client with the correct credentials can decrypt it. The existing scheme of hidden credentials requires that the server grant access to the encrypted resource directly to the client during the negotiation without knowing whether or not the client can decrypt it. It would be a burden if the resources were very large. We found that when the server grants access to services rather than resources, the existing hidden credentials schemes are insecure under our policy attacks*, since the server can illegally learn the client's credentials from the attack. In this paper, we propose a scheme to stop the server from mounting a policy attack.*

1. Introduction

In Trust Negotiation, two strange parties can exchange digital credentials that contain some attributes of information for access control. Hidden Credential stems from the paradigm of Trust Negotiation [6, 7, 8], which guards sensitive resources by attribute-based policies that can be fulfilled by publicly verifiable digital credentials issued by some third party.

Conceptually, a trust negotiation problem is given as follows. Let us denote by Alice and Bob the participants, where Alice is the client and Bob is the server. Bob grants access to sensitive resources to his clients who have the correct credentials. Because of the sensitivity, Bob does not want to reveal his policies in order to protect his sensitive resources. Alice has a correct credential, but she doesn't want to disclose her credential to Bob. Some recent works [2, 3, 9] have performed this type of attribute-based access controls for protecting Alice's credentials and Bob's policies [5]. Based on identity-based encryption (IBE) in [1], it is not hard to achieve hidden credentials.

In hidden credential, Bob encrypts a resource in such a way that Alice can decrypt it if she has the right credentials [2], where the encryption key is Bob's policy and the decryption key is Alice's credential. There are three key properties in a hidden credential system: (1) Protection of Alice's sensitive credentials. Bob never sees Alice's credential and never knows whether Alice can access the resource. (2) Protection of Bob's sensitive policies. When Bob encrypts his resource that Alice needs, Alice can decrypt it if her credential matches one of Bob's policies. If not, she will learn nothing about Bob's resource (3) Protection of Bob's sensitive resources. The encrypted resource is not usable if one does not have the right credential; therefore, the resources are hidden safely.

1.1. Related Work and Analysis

Based on Boneh-Franklin's IBE [1], Holt *et al* introduced the notion of hidden credentials [2]. They gave a formal description of hidden credentials and gave an application of hidden credentials. Frikken *et al* later improved computational efficiency of hidden credential decryption and enhanced security with a secret splitting scheme [9].

The existing schemes of hidden credentials can protect privacy of both Alice and Bob. Assuming that Bob does not grant access to services but resources, Bob never

*This work is partially supported by a grant from the Fund of National Natural Science Foundation of China (#60502047), Education Bureau of Fujian Province (#JB05329)

1-4244-0605-6/06/$20.00 ©2006 IEEE.

knows whether Alice can access to resources she requested. Frikken *et al* showed this drawback in [4]. In practice, the server could grant access to services rather than resources. It means that the server determines whether a client can access to a resource or not but doesn't transmit the resource to the client directly.

In hidden credentials, Alice can protect her credentials and Bob can protect his policies. However, it could be the case that Alice would only learn part of policies if she had decrypted the ciphertext from Bob. In [4], the situation is improved that Alice cannot learn Bob's credentials and policies and Bob can not learn Alice's credentials and policies (in the case of mutual negotiation). Furthermore, Alice can not learn any policy even if she has accessed the resource, because which credential satisfies which Bob's policy is indistinguishable to her.

Although the scheme in [4] offers better privacy protection than the other schemes of hidden credentials, we found that Alice can learn Bob's policies when she requests to access a resource repeatedly. For example, let Alice's credentials be $\{C_1, C_2, C_3, C_4\}$ and she can access a resource with the credentials in the first trust negotiation, but she does not know which credential satisfies Bob's policy. Then, Alice can request another access to the same resource with a subset of $\{C_1, C_2, C_3, C_4\}$, if she can access the resource with $\{C_1, C_2\}$ and $\{C_2, C_3\}$ but not with $\{C_1, C_3\}$, Alice learns that the credential C_2 must satisfy Bob's policy, therefore Bob's policy leaks.

Based on the scenario that Bob grants services, there will be another private problem we have to address. If Alice can access a resource, Bob learns that her credentials satisfy his policies (does not which one). Alice's credentials can be hidden from Bob when Bob's policies are a large set. For a small set of policies, Bob will then have a better chance to guess Alice's credentials. Therefore, to mount an attack, Bob can set his policies with a lot of random strings such that they are indistinguishable to Alice. Actually, there could be only one Bob's policy matches with a credential. If Alice had accessed the resource with the correct credential, she would reveal her credential to Bob (so-called perilous policy attacks). For example, Bob can set his policies $P_{Bob} = P_1 \vee P_2 \vee P_3$ which matches with the credential C_1, C_2, or C_3, but the credentials C_2 and C_3 do not exist at all (according to Bob's setting). If Alice can access to the resource, Bob learns that Alice must possess the credential C_1.

We found that the flaw of the schemes in [2, 9, 4] is due to the fact that Alice cannot verify Bob's policies. Therefore, we propose an approach of zero-knowledge proof to Bob's policies, where Alice can verify whether Bob has mounted a policy attack but knows nothing about Bob's policies.

1.2. Our Contributions

We propose a novel scheme of hidden credentials which can protect Alice's credentials from policy attacks in the following scenarios: (1) Alice's credentials can be hidden: Bob never learns her credentials accurately but only knows her credentials satisfy his policies. (2) Bob's policies can be hidden: Alice never learns his policies without right credentials but only knows part of Bob's policies which her credentials satisfy. (3) User Anonymity: Bob does not know who accesses a resource. (4) Service-oriented: Bob grants access to services and knows whether a client can access the corresponding resource or not.

2. Definitions and Notations

Bilinear Pairing. Let $\mathbb{G}_1$ be (additive) cyclic group of prime order q. Let P, Q be a generator of $\mathbb{G}_1$. A map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ (here $\mathbb{G}_2$ is a multiplicative group such that $|\mathbb{G}_1| = |\mathbb{G}_2| = q$) is called a bilinear pairing if this map satisfies the following properties: (1) Bilinear: for all $P, Q \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_p$, we have $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$. (2) Non-degeneracy: $\hat{e}(P, P) \neq 1$. In other words, if P be a generator of $\mathbb{G}_1$, then $\hat{e}(P, P)$ generates $\mathbb{G}_2$. (3) Computability: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for all $P, Q \in \mathbb{G}_1$.

ID-Based Encryption (IBE).

Setup. Choose a random generator $P \in \mathbb{G}_1$. Pick a random $s \in \mathbb{Z}_q^*$ and set $P_{pub} = sP$. Choose four cryptographic hash functions $H_1, \cdots, H_4$. $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$; $H_2 : \mathbb{G}_2 \rightarrow \{0,1\}^n$; $H_3 : \{0,1\}^n \times \{0,1\}^n \rightarrow \mathbb{Z}_q^*$; $H_4 : \{0,1\}^n \rightarrow \{0,1\}^n$. The message space is $M = \{0,1\}^n$. The parameters are $params = \langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, H_1, H_2.H_3, H_4\rangle$. The master-key is s.

Extract. For a given string $ID \in \{0,1\}^*$, compute $Q_{ID} = H_1(ID) \in \mathbb{G}_1$, set the private key d_{ID} to be $d_{ID} = sQ_{ID}$ where s is the master-key.

Encrypt. To encrypt $M \in \{0,1\}^n$ under the public key ID, compute $Q_{ID} = H_1(ID) \in \mathbb{G}_1$, choose a random $\sigma \in \{0,1\}^n$, set $r = H_3(\sigma, M)$, and set the ciphertext to be$\langle U, V, W\rangle = \langle rP, \sigma \oplus H_2(g_{ID}^r), H_4(\sigma) \oplus M\rangle$ where $g_{ID} = \hat{e}(Q_{ID}, P_{pub}) \in \mathbb{G}_2$.

Decrypt. Let $\langle U, V, W\rangle$ be a ciphertext encrypted using the public key ID. If $U \notin \mathbb{G}_1^*$, abort. To decrypt M with the private-key $d_{ID} \in \mathbb{G}_1$, compute $V \oplus H_2(\hat{e}(d_{ID}, U)) = \sigma$, compute $W \oplus H_4(\sigma) = M$, and set $r = H_3(\sigma, M)$. Check if $U = rP$. If not, abort. Output M as the decryption of $\langle U, V, W\rangle$.

Policy and Credential. Using ID-Based Encryption, ID is denoted as a Policy and private key d_{ID} as a Credential. In

our paper, we use P_i to denote a policy and C_i a corresponding credential.

Simple Policy. A simple policy consists of a set of attributes. For example, an attribute could be "president" or "dean". If a resource is encrypted with a simple policy, the ciphertext can be decrypted with an associated credential matching the policy.

Complex Policy. A complex policy consists of multiple simple policies with monotonic Boolean function $\vee$ and $\wedge$. For example, A complex policy could be defined as $P_{Bob} = P_1 \vee (P_2 \wedge P_3) \vee (P_4 \wedge P_5)$. In order to access Bob's resources, Alice's credentials must include $\{C_1\}$, $\{C_2, C_3\}$, or $\{C_4, C_5\}$, which match Bob's policies. For example, if Alice has credentials $\{C_1, C_2\}$, she will be granted with access to the corresponding resource, but rejected if she only has credentials $\{C_2, C_4\}$.

One-Time Password (OTP). The OTP is randomly chosen by Bob and can be used only once. If a client requests to access a resource, Bob will randomly choose a OTP encrypted with his policies. If the client can send the OTP back to Bob, the client is granted with an access to the resource.

Simple Encryption (SE). The ciphertext of SE is the tuple: $\langle U, V, W\rangle = \langle rP_{gene}, \sigma \oplus H_2(g_{p_i}^r), H_4(\sigma) \oplus M\rangle$ where P_{gene} is a generator of $\mathbb{G}_1$ and M is a message that contains OTP. The ciphertext is encrypted with policy P_i and can only be decrypted with C_i. So, if Bob's policy is $P_{Bob} = P_{president}$, Alice will reveal her credential $C_{president}$ when she can decrypt the ciphertext.

Complex Encryption (CE). For $P_{Bob} = (P_1 \wedge P_2) \vee P_3 \vee P_4$, the ciphertext of CE is the tuple: $\langle U, V_1, V_2, V_3, W\rangle = \langle rP_{gene}, \sigma \oplus H_2(g_{p_1}^r) \oplus H_2(g_{p_2}^r), \sigma \oplus H_2(g_{p_3}^r), \sigma \oplus H_2(g_{p_4}^r), H_4(\sigma) \oplus M\rangle$. M is encrypted with diversity of policies and can be decrypted with diversity of credentials too. The difference between SE and CE is that r in later is not set by $r = H_3(\sigma, M)$ but randomly chosen by Bob [9].

In CE, Alice decrypts M from one of V_i, therefore her credentials are indistinguishable to Bob if he encrypted OTP honestly. However, in practice Bob must be able to prove security against a policy attack.

Trust Parameters (TP). Each TP has a parameter and a signature which can prove that the policy in the parameter is valid. The parameter is set by $g_{P_i}^{r_0} = \hat{e}(Q_{P_i}, r_0 P_{pub}) = \hat{e}(Q_{P_i}, P_{pub})^{r_0}$, where r_0 is secretly chosen by Bob; the signature is set by $Sgn(H_2(g_{P_i}^{r_0}))$. We denote by $TP(P_i, r_0)$ the parameter and signature of policy P_i with r_0. So, if Bob shows $TP(P_i, r_0)$ to Alice, she will learn nothing but the policy is valid (see section 5 for detail analysis).

Notations. Denote by $\mathbf{PKG}$ the trust third party who sets all policies $\mathbf{P_{PKG}}$ and issues all credentials and TP. Denote by $\mathbf{D_{Sgn}}$ the public key which can verify TP. For example, if Bob shows $TP(P_i, r_0)$ to Alice, she can verify $H_2(\hat{e}(Q_{P_i}, P_{pub})^{r_0}) \stackrel{?}{=} D_{sgn}(TP(P_i, r_0))$. Denote by σ^* a fix bit and $\sigma^* = 1^t$ for some constant t. It is used to help Alice make a judgement of correct decryption. Denote by $\mathbf{R_{Bob}}$ an OTP randomly chosen by Bob. Denote by $\mathbf{S_{TP}}$ the intersection of two sets. One is computed by Bob through his policies, the other is computed by Alice through TP. Denote by $\mathbf{N_{Alice}}$ the minimum numbers of valid policies, in which Alice can ensure that her credentials are indistinguishable to Bob.

3. Our Scheme

In this section, we present our novel scheme that is secure against the policy attack.

Parameter Phase:

PKG broadcasts the *params* to all users. $params = \langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, t, n, P_{gene}, P_{pub}, \sigma^*, D_{Sgn}, H_1, H_2, H_3, H_4\rangle$. The *params* is the same as in the IBE. In addition, we pick $\sigma^* = 1^t$ and a public key D_{Sgn} for verifying TP. The user transmits his/her identity credential and P_i to the PKG in security. The PKG verifies the credential and P_i. If the user qualifies to possess C_i, the PKG computes $Q_{P_i} = H_1(P_i) \in \mathbb{G}_1$, $C_i = sQ_{P_i}$ and transmits C_i to the user.

Trust Parameters:

Bob computes $r_0 P_{pub}$ and transmits $r_0 P_{pub}, P_1, P_2, \cdots, P_k$ to the PKG, where $P_1, P_2, \cdots, P_k$ are Bob's policies. The PKG checks $P_1, P_2, \cdots, P_k \stackrel{?}{\in} P_{PKG}$. If $P_1, P_2, \cdots, P_k$ are valid, the PKG computes $TP(P_i, r_0), i \in \{1, 2, \cdots, k\}$ and transmits them to Bob.

Request:

Alice makes an access "*request*" to Bob. If Bob's policies are $P_{Bob} = P_1 \vee P_2 \cdots \vee P_k$, Bob computes $HC_E(P_{Bob})$: Compute $Q_{P_i} = H_1(P_i) \in \mathbb{G}_1, i \in \{1, 2, \cdots, k\}$. Choose a random $\sigma' \in \{0,1\}^{n-t}$ and set the σ with $\sigma = \sigma^* \| \sigma' \in \{0,1\}^n$. Compute $r = H_3(\sigma, M)$. Compute $g_{P_i} = \hat{e}(Q_{P_i}, P_{pub}) \in \mathbb{G}_2, i \in \{1, 2, \cdots, k\}$. Set ciphertext $HC_E(P_{Bob}) = \langle rr_0 P_{gene}, \sigma \oplus H_2(g_{p_1}^{rr_0}), \sigma \oplus H_2(g_{p_2}^{rr_0}), \cdots, \sigma \oplus H_2(g_{p_k}^{rr_0}), H_4(\sigma) \oplus M\rangle$, where M are R_{Bob} and $TP(P_i, r_0), i \in \{1, 2, \cdots, k\}$.

Alice receives the ciphertext $HC_E(P_{Bob}) = \langle U, V_1, V_2, \cdots, V_k, W\rangle$ from Bob. Assume that Alice has credential C_m. Alice does the following: Compute $\hat{e}(C_m, U) = \hat{e}(C_m, rr_0 P_{gene}) = \hat{e}(Q_{P_m}, P_{gene})^{rr_0 s} = \hat{e}(Q_{P_m}, P_{pub})^{rr_0} = g_{P_m}^{rr_0}$. Compute $V_1 \oplus H_2(g_{P_m}^{rr_0}) = \sigma$. If $\sigma^* \in \sigma$, then compute $W \oplus H_4(\sigma) = M$ and output M as the decryption of the $HC_E(P_{Bob})$; else go to (2) to compute $V_2 \oplus H_2(g_{P_m}^{rr_0}) = \sigma$ with V_2, until V_k.

If Alice cannot output M as decryption of $HC_E(P_{Bob})$, she has to stop negotiation. Bob then is unable to receive R_{Bob} from Alice and stops negotiation. Consequently, Al-

ice fails to access to the resource; otherwise, Alice continues the verification process.

Alice does the following: Verify $\hat{e}(Q_{P_i}, P_{pub})^{r_0}, i \in \{1, 2, \cdots, k\}$ are valid with $TP(P_i, r_0)$ and D_{PKG}. Compute $\left(\hat{e}(Q_{P_i}, P_{pub})^{r_0}\right)^r, i \in \{1, 2, \cdots, k\}$ with r. Compute $\sigma \oplus H_2\left(\hat{e}(Q_{P_i}, P_{pub})^{rr_0}\right), i \in \{1, 2, \cdots, k\}$. Compute the intersection of two sets: $S_{TP} = \{\sigma \oplus H_2\left(\hat{e}(Q_{P_i}, P_{pub})^{rr_0}\right), i \in \{1, 2, \cdots k\}\} \cap \{V_1, V_2, \cdots V_k\}$. If $|S_{TP}| < N_{Alice}$, stop the negotiation; else, continue the process.

Response Phase:
Alice gets R_{Bob} from M and transmit it to Bob. Bob checks whether R_{Bob} is correct or not. If not, Bob will reject Alice from accessing the resource, else she is granted with the corresponding resource.

With no decryption effort at all, any client(without credentials) knows how many policies in $HC_E(P_{Bob})$ from $\langle V_1, V_2, \cdots, V_k\rangle$. Bob can use dummy policy to hide his policies. For instance, if $P_{Bob} = P_1 \vee P_2$, he can encrypt $HC_E(P_{Bob})$ with $P_{Bob} = P_1 \vee P_2 \vee P_{*3} \vee P_{*4}$ where P_{*i} are not exist at all.

Trust Parameters can be used repeatedly. The policy in $TP(P_i, r_0)$ is hidden from Alice (we will analyse it in section 5). So, Bob can get TP from the PKG only once and use it repeatedly.

4. Security Analysis

We present three computational hard problems here on which the security of IBE and our scheme is based.

(1) Discrete Logarithm Problem (DL). Given two elements $P, Q \in \mathbb{G}_1$,find an integer $\alpha \in \mathbb{Z}_q^*$ such that $Q = \alpha P$ whenever such an integer exists.

(2) Computational Diffie-Hellman Problem (CDH). Given $\langle P, aP, bP\rangle$ for some unknown $a, b \in \mathbb{Z}_q^*$ where P is a generator of $\mathbb{G}_1$, compute abP.

(3) Bilinear Diffie-Hellman Problem (BDH). Given $\langle P, aP, bP, cP\rangle$ for some unknown $a, b, c \in \mathbb{Z}_q^*$ where P is a generator of $\mathbb{G}_1$, compute $W = \hat{e}(P, P)^{abc} \in \mathbb{G}_2$.

Bob's policies are hidden from Alice. Even if Alice can decrypt the $HC_E(P_{Bob})$ with C_m and knows that $P_m \in P_{Bob}$ and $P_m \in TP(P_i, r_0)$. Let $g = \hat{e}(Q_{P_m}, P_{pub})$ and $h = \hat{e}(Q_{P_m}, P_{pub})^{r_0}$, the CDH problem holds. So r_0 is hidden to Alice.

For $\{P, Q_{P_s}, P_{pub}, r_0 P\}$ ($P_s$is a policy supposed by Alice), the BDH problem holds. So, Alice cannot compute $\hat{e}(Q_{P_s}, P_{pub})^{r_0}$ and verify $g_{P_s}^{r_0} \stackrel{?}{\in}$ TP.

With σ and r, Alice can verify $\sigma \oplus H_2(g_{P_s}^{rr_0}) \stackrel{?}{\in} \{V_1, V_2, V_3, \cdots, V_k\}$ iff she can compute $g_{P_s}^{r_0}$. From the above, We know that Alice will fail to verify.

Our scheme is secure against Bob's policy attacks (Alice's credentials are hidden from Bob). In our scheme, how many policies are valid in $HC_E(P_{Bob})$ can be learned from S_{TP}, which is computed by TP. Because Bob cannot get TP of dummy policy, then it is impossible for Bob to mount an attack from policies. Alice's credentials are hidden from Bob for that she can learn S_{TP} and decides to continue or not.

Because of the space restriction, we will provide security proofs in the full version of this paper.

5. Conclusion

We showed how a perilous attack in hidden credentials from the Bob's policy will reveal Alice's sensitive credentials. We proposed a novel scheme that is secure against the policy attack, using trust parameters. In our scheme, Alice can find whether Bob launched an attack and then decide if the resource access should be granted.

References

[1] D. Boneh and M. Franklin. *Identity-Based Encryption from the Weil Pairing.* In J. Kilian, editor,Advances in Cryptology-Crypto01, Springer-Verlag, LNCS 2139, pp.213-229, 2001.

[2] J. E. Holt, R. W. Bradshaw, K. E. Seamons, and H. Orman. *Hidden credentials.* In Proceedings of the 2nd ACM Workshop on Privacy in the Electronic Society, Oct. 2003.

[3] K.B.Frikken, M.J.Atallah, and J.Li. *Hidden access control policies with hidden crentials.* In Proceedings of the 3rd ACM Workshop on Privacy in the Electronic Society,Oct.2004

[4] K. Frikken, J. Li, and M. Atallah. *Trust Negotiation with hidden credentials, hidden policies, and policy cycles.* In To appear in NDSS 2006, 2006.

[5] A. Menezes, T. Okamoto, S. Vanstone *Reducing elliptic curve logarithms to logarithms in a finite field.* IEEE Tran. on Info. Th.Vol. 39, pp. 1639-1646, 1993.

[6] W. H. Winsborough and N. Li. *Protecting Sensitive Attributes in Automated Trust Negotiation.* Proceedings of ACM Workshop on Privacy in the Electronic Society, Washington, DC, November 2002.

[7] W. H. Winsborough and N. Li. *Towards Practical Trust Negotiation.* Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks (POLICY 2002), Monterey, California, June 2002.

[8] W.H. Winsborough, K. E. Seamons, and V. E. Jones. *Automated Trust Negotiation.* DARPA Information Survivability Conference and Exposition, Hilton Head, SC, January 2000.

[9] R. Bradshaw, J. Holt, and K. Seamons. *Concealing Complex Policies with Hidden Credentials.* In Proceedings of ACM CCS. 2004.