The Oblivious Commitment-Based Envelope Protocols

Concerning Crypto, Communications and Digital Identity Management

Ning Shang

Purdue University

January 23, 2009

Prologue

A conversation between Alice and Bob

Alice: I know how to solve the discrete logarithm problem $c = g^x$ for x.
Bob: Show me.

Prologue

A conversation between Alice and Bob

Alice: I know how to solve the discrete logarithm problem $c = g^x$ for x.
Bob: Show me.

Case 1: an easy way

- Alice shows x to Bob.
- Bob verifies $c = g^x$, thus is convinced.
- Bob immediately claims: I know how to solve the discrete logarithm problem $c = g^x$!!!
- Alice is not very happy, because Bob is as knowledgeable as she is.

Prologue

A conversation between Alice and Bob

Alice: I know how to solve the discrete logarithm problem $c = g^x$ for x.
Bob: Show me.

Case 1: an easy way

- Alice shows x to Bob.
- Bob verifies $c = g^x$, thus is convinced.
- Bob immediately claims: I know how to solve the discrete logarithm problem $c = g^x$!!!
- Alice is not very happy, because Bob is as knowledgeable as she is.

Case 2: ZKPK

- Alice chooses a random $y \in \{1, \ldots, p-1\}$, computes $d = g^y$, and sends Bob d.
- Bob sends Alice a random challenge $e \in \{1, \ldots, p-1\}$.
- Alice computes $u = y + e \cdot x$, and send Bob u.
- Bob verifies $g^u = d \cdot c^e$, and is convinced.
- Alice is still happy, because she still knows more than Bob.

Prologue

Another conversation between Alice and Bob

Alice: I know the values x and r such that $M = g^x h^r$, and $x \geq 2009$.
Bob: show me.

Prologue

Another conversation between Alice and Bob

Alice: I know the values x and r such that $M = g^x h^r$, and $x \geq 2009$.
Bob: show me.

- Lesson learned from Case 1, Alice is not willing to show x and r to Bob.
- Alice performs a ZKPK as in Case 2. But this does not convince Bob that $x \geq 2009$.

Prologue

Another conversation between Alice and Bob

Alice: I know the values x and r such that $M = g^x h^r$, and $x \geq 2009$.
Bob: show me.

- Lesson learned from Case 1, Alice is not willing to show x and r to Bob.
- Alice performs a ZKPK as in Case 2. But this does not convince Bob that $x \geq 2009$.

Alice stays happy

Alice can make use of the Oblivious Commitment-Based Envelope (OCBE) protocols to convince Bob, at the same time having her secret (x and r) kept secret.

OCBE Protocols

[Reference] *OACerts: Oblivious Attribute Certificates* by J. Li et al.

OCBE: Parties in communications

- T - trusted third party:
  publishes system parameters $g, h \in G$, where G is a finite group of p elements. T also published its public key for digital signature scheme.
- P - principal (user):
  holds a Pedersen commitment $M = g^x h^r$, with x being the committed value
- SP - service providers:
  makes access control policy on x

Assumption: The content of the service SVC provided by SP is encrypted using a symmetric algorithm with key SK.

OCBE Protocols

OCBE protocols solve the following problem

P makes a request for service SVC from SP. SP sends encrypted service content to P. The service can be correctly received by P if and only if P satisfies the condition Cond, specified in the policy of SP, without P showing the details in clear.

EQ-OCBE

Case Cond = "$x = x_0$".

Before execution of the protocol, SP and P agree on a symmetric encryption algorithm $\mathcal{E}$ and a cryptographic hash function $H(\cdot)$.

After verifying the validity and ownership (e.g. via ZKPK) of $M = g^x h^y$, T signs M and hands M together with the signature to P.

P requests service SVC from SP.

SP tells P the expected condition "$x = x_0$".

P shows M and its signature signed by T.

1. SP picks $z \in \mathbb{Z}_p^{\times}$, computes $\delta = (Mg^{-x_0})^z$, and then sends to P the pair $(\eta = h^z, C = \mathcal{E}_{H(\delta)}[SK])$.

2. Upon receiving (η, C) from SP, P computes $\delta' = \eta^y$, and decrypts C using symmetric encryption key $H(\delta')$.
If $x = x_0$, SK can be successfully recovered from C.

Example: case of equality

G: finite group of order p (large); $g, h \in G$
$H(\cdot)$: SHA-1; $\mathcal{E}$: AES
Encode "STATE = IN(14)" as "$x = 14$".

1. An Indiana resident P requests service from SP. SP sends its policy $\{\text{STATE} = \text{IN}(14)\}$ to P. After receiving the policy, P sends to SP its commitment $M = g^{14}h^{1234}$ signed by T. Note the value "1234" is known only to P.
2. SP picks random secret $z = 5678$, computes $\delta = (Mg^{-14})^z = (g^{14}h^{1234}g^{-14})^{5678} = (h^{1234})^{5678}$. SP sends to P the pair
$$(\eta = h^{5678}, C = \mathcal{E}_{H((h^{1234})^{5678})}[SK]).$$
3. P computes $\delta' = \eta^{1234} = h^{5678 \cdot 1234} = \delta$ and decrypts C using the key $H(\delta')$.

GE- and LE-OCBE

Case Cond $= x \in [a, b]$

Before execution of the protocols, SP and P agree on a symmetric encryption algorithm $\mathcal{E}$, and three cryptographic hash functions H, $\hat{H}$ and H'. SP chooses two secrete values SK_1 and SK_2, and sets the encryption key for the content of service SVC to be

$$SK = \hat{H}(SK_1 || SK_2).$$

GE-OCBE: condition $x \geq a$

1. T chooses a positive integer ℓ so that the bit length of $(b - a)$ is less than ℓ, and that $2^\ell < p/2$, where p is still the order of the group G.

2. After verifying the validity and ownership of the commitment $M = g^x h^r$ from P, T signs it and sends the signature to P.
P requests service SVC from SP.
SP tells P the expected condition $''x \geq a''$.
P shows M and its signature signed by T.

3. P computes $d = x - a \pmod{p}$. P picks random $r_1, \ldots, r_{\ell-1}$, and sets $r_0 = r - \sum_{i=1}^{\ell-1} 2^i r_i \pmod{p}$. Let $(d_{\ell-1} \ldots d_1 d_0)_2$ be the binary representation of d. P computes commitments $M_i = g^{d_i} h^{r_i}, i = 0, \ldots, \ell - 1$, then sends them to SP.

4. SP verifies that $Mg^{-a} = \prod_{i=0}^{\ell-1}(M_i)^{2^i}$.
SP randomly chooses ℓ secret values $k_0, \ldots, k_{\ell-1}$ and sets the encryption key $k = H'(k_0 || \ldots || k_{\ell-1})$.
SP picks $y \in \mathbb{Z}_p^{\times}$, and computes $\eta = h^y$ and the encrypted information, $I_1 = \mathcal{E}_k[SK_1]$, where SK_1 is half of the information needed to derive the encryption/decryption key, SK, for the requested service SVC. The other half of the information will be obtained from process for condition $x \leq b$.
SP computes $\delta_i^0 = (M_i)^y, \delta_i^1 = (M_i/g)^y$, $C_i^0 = H(\delta_i^0) \oplus k_i$, and $C_i^1 = H(\delta_i^1) \oplus k_i$, for $0 \leq i \leq \ell - 1$.
SP sends to P the tuple $(\eta, C_0^0, C_0^1, \ldots, C_{\ell-1}^0, C_{\ell-1}^1, I_1)$

5. Upon receiving the tuple $(\eta, C_0^0, C_0^1, \ldots, C_{\ell-1}^0, C_{\ell-1}^1, I_1)$ from SP, P computes $\delta_i' = \eta^{r_i}$, and $k_i' = H(\delta_i') \oplus C_i^{d_i}$.
P then computes $k' = H'(k_0' || \ldots || k_{\ell-1}')$, then decrypts C with the key k'.

We have $k' = k$ and thus P can successfully decrypt I_1 using k', if and only if $x \geq a$.

Example: inequality $\geq$

Costumer P has a receipt from a previous purchase of $x = \$83$. The service provider SP issues policy that the it must be $x \in [70, 100]$ that P can receive service.

As before,

G: finite group of order p (large); $g, h \in G$

$H(\cdot)$: SHA-1; $\mathcal{E}$: AES

We show how P can be served without revealing the value x in clear to SP.

1. T chooses $\ell = 8$ $(2^{\ell} << p)$

2. T signs P's commitment $M = g^{83}h^{1234}$, where the value "1234" is known only to P.

3. P computes $d = 83 - 70 = (00001101)_2$.
P randomly picks $r_1 = 1, r_2 = 2, \ldots, r_7 = 7$ and sets
$r_0 = 1234 - 28 = 1206$.
P computes commitments $M_i = g^{d_i}h_{r_i}$, where d_i is the ith bit of d. So we have $M_0 = gh^{1206}, M_1 = h, M_2 = gh^2$, and so on. P sends all M_i to SP.

4. SP verifies $Mg^{-70} = \prod_{i=0}^{\ell-1}(M_i)^{2^i}$, then chooses random $k_0 = 01, k_1 = 12, k_2 = 23, \ldots, k_7 = 78$ and sets

$$\begin{aligned} k &= H('0112233445566778') \pmod{p} \\ &= 0x7366995735b395af1c22683ef7219347a8a0899c \pmod{p} \end{aligned}$$

SP picks $y = 5678$, computes $\eta = h^{5678}$, and does encryption $I_1 = \mathcal{E}_k[SK_1]$.
SP computes
$\delta_i^0 = (M_i)^{5678} = (g^{d_i} h^{r_i})^{5678}, \delta_i^1 = (M_i g^{-1})^{5678} = (g^{d_i - 1} h^{r_i})^{5678}$, one of which is $h^{r_i \cdot 5678}$.
SP computes $C_i^0 = H(\delta_i^0) \oplus k_i, C_i^1 = H(\delta_i^1) \oplus k_i$.
SP sends to P the tuple

$$(\eta = h^{5678}, C_0^0, C_0^1, \ldots, C_7^0, C_7^1, I_1).$$

5. P computes $\delta_i' = \eta^{r_i} = h^{5678 \cdot r_i}$. Then depending on d_i, P chooses $C_i^{d_i}$ to XOR with δ_i' to obtain $k_i' = k_i$.
P computes $k' = H(k_0' || \ldots || k_7') = H('0112233445566778') = k$.
P now can decrypts I_1 using $k' = k$.

LE-OCBE: condition $x \leq b$

Similar to the case of $x \geq b$.

1. T chooses a positive integer ℓ so that the bit length of $(b - a)$ is less than ℓ, and that $2^\ell < p/2$, where p is still the order of the group G.

2. After verifying the validity and ownership of the commitment $M = g^x h^r$ from P, SP signs it and sends the signature to P.
P requests service SVC from SP.
SP tells P the expected condition $''x \leq b''$.
P shows M and its signature signed by T.

3. P computes $d = b - x \pmod p$. P picks random $r_1, \ldots, r_{\ell-1}$, and sets $r_0 = -r - \sum_{i=1}^{\ell-1} 2^i r_i \pmod p$. Let $(d_{\ell-1} \ldots d_1 d_0)_2$ be the binary representation of d. P computes commitments $M_i = g^{d_i} h^{r_i}, i = 0, \ldots, \ell - 1$, then sends them to SP.

4. SP verifies that $M^{-1}g^b = \prod_{i=0}^{\ell-1}(M_i)^{2^i}$.
SP randomly chooses ℓ secret values $k_0, \ldots, k_{\ell-1}$ and sets the encryption key $k = H'(k_0||\ldots||k_{\ell-1})$.
SP picks $y \in \mathbb{Z}_p^{\times}$, and computes $\eta = h^y$ and the encrypted information $I_2 = \mathcal{E}_k[SK_2]$, where SK_2 is the other half of the information for retrieving the actual enc/dec key for SVC.
SP computes $\delta_i^0 = (M_i)^y, \delta_i^1 = (M_i/g)^y, C_i^0 = H(\delta_i^0) \oplus k_i$, and $C_i^1 = H(\delta_i^1) \oplus k_i$, for $0 \leq i \leq \ell - 1$.
SP sends to P the tuple $(\eta, C_0^0, C_0^1, \ldots, C_{\ell-1}^0, C_{\ell-1}^1, I_2)$

5. Upon receiving the tuple $(\eta, C_0^0, C_0^1, \ldots, C_{\ell-1}^0, C_{ell-1}^1, I_2)$ from SP, P computes $\delta_i' = \eta^{r_i}$, and $k_i' = H(\delta_i') \oplus C_i^{d_i}$.
P then computes $k' = H'(k_0'|| \ldots ||k_{\ell-1}')$, then decrypts I_2 with the key k'.

We have $k' = k$ and thus P can successfully decrypt I_2 using k', if and only if $x \leq b$.

Combine the two: $x \in [a, b]$

Now P computes the enc/dec key SK for service SVC as follows:

$$SK = \hat{H}(SK_1 || SK_2).$$

Agg-EQ-OCBE

Efficient treatment of multiple equality conditions

Problem: how to enforce policy
{STATE=IN(14) -AND- SCHOOL=Purdue(56)}
with the cost of one computation?

We need to use a cryptographic hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^\times$.

1. Set commitment $M_i = g^{H(x_i)} h^{y_i}, i = 1, \ldots, n$ for condition $x_i = a_i$, and aggregate commitment $M = \prod_{i=1}^n M_i$
2. Perform QE-OCBE for condition $x = \sum_{i=1}^n H(a_i)$ on aggregate commitment M.

Use assumption that it is hard to find $\tilde{x_1}', \ldots, \tilde{x_n}'$ such that

$$A = \sum_{i=1}^{n} H(\tilde{x_i}')$$

for a given A.

Application of OCBE

Privacy-Preserving Management of Transactions' Receipts for Mobile Environments, F. Paci et al. uses OCBE protocols for attribute-based identity management.

Prototypes for mobile phone and PC are developed.

Epilogue

Let's go back to an earlier conversation

Another conversation between Alice and Bob

Alice: I know the values x and r such that $M = g^x h^r$, and $x \geq 2009$.
Bob: show me.

Alice's solution

- Bob chooses a random bit string message.
- Alice and Bob performs a GE-OCBE protocol for $x_0 = 2009$, with message encrypted and transfered.
- Alice decrypts and shows message to Bob.
- Bob verifies message he receives is indeed the one of his original pick, thus is convinced.
- Alice lives happily ever after.