

Multiple Hierarchy Wildcard Encryption

Christopher A. Seaman¹, Kent D. Boklan², and Alexander W. Dent³

¹ Graduate Center, City University of New York, USA

² Queens College, City University of New York, USA

³ Royal Holloway, University of London, UK

1 Non-Selective Identity Security in the Random Oracle Model

Like the Boneh-Boyen HIBE it is based on [BB04], the multi- TA scheme is only proven secure in the (selective-identity) multi- TA IND-sWID-CPA model. For non-wildcard single TA schemes, selective-identity IND-s(H)ID security in the standard model can be transformed into non-selective-identity secure IND-(H)ID schemes in the random oracle model [BBG05]. This technique was extended to securely enable wildcards in IND-sHID-CPA WIBE schemes [ACD⁺06] and we further extend it to prove non-selective-identity security of our proposed multi- TA scheme in the random oracle model.

The security game in the random oracle model is the same as in the standard model with the addition of one new oracle [BR93]. This oracle responds to each query with a uniformly random element chosen from its output domain. Additionally, for each specific query the response is always the same whenever that query is made.

To prove security in a non-selective-identity setting we alter the routines of the multi- TA WIBE scheme to use the hash of an identity ($H_i(ID_i)$) rather than the identity (ID_i) for each level i . The key derivation and encryption routines pass the identity to a hash function (modeled as a random oracle) and use the result in place of the identity in the relevant algorithm. This hash function maps from the finite space of identifiers to an equal or smaller sized target space. This transform changes the pattern $\mathbf{P} = (P_1, P_2, \dots, P_\ell)$ into pattern $H(\mathbf{P}) = \mathbf{P}' = (P'_1, P'_2, \dots, P'_\ell)$ as follows:

$$P'_i = H_i(P_i) \text{ for } i \notin W(\mathbf{P}), \text{ otherwise } H(P_i) = * = P'_i$$

We note that a finite family of independent random oracles of predetermined maximum size may be simulated using a single random oracle. Given a single random oracle $H(\text{query})$, a WIBE of maximum depth L and a fixed-length format to enumerate $i \in \{1 \dots L\}$, we may create i independent random oracles. Each oracle $H_i(\text{query})$ corresponding to level i of the WIBE is simulated by prepending the query with the number of the oracle it is addressed to, $H_i(\text{query}) = H(i||\text{query})$.

Theorem 1. *Suppose that there exists a polynomial-time attacker \mathcal{A} against the non-selective-identity multiple TA IND-WID-CPA Boneh-Boyen WIBE with advantage ϵ with access to q_H queries from each of the L random oracles associated with hierarchy levels, then there exists a polynomial-time attacker \mathcal{B} against the selective-identity multiple TA IND-sWID-CPA Boneh-Boyen WIBE with advantage ϵ' when allowed q_K key derivation queries with ϵ' bounded:*

$$\epsilon' \geq \left(\frac{\epsilon}{2^n L (q_H + 1)^L} \right) \left(1 - \frac{(q_H + 1)}{|ID|} \right)^L$$

Proof Suppose there exists adversary \mathcal{A} against the multi- TA IND-WID-CPA scheme (with hashed identities), we will construct an adversary \mathcal{B} which gains an advantage against the multi- TA IND-sWID-CPA scheme using the algorithm \mathcal{A} as a black box.

For a multi- TA WIBE of maximum depth L consisting of a maximum of k TA 's, the algorithm \mathcal{B} runs as follows:

1. \mathcal{B} randomly chooses a length $\ell^* \in \{1 \dots L\}$ for its challenge identity vector. Also, \mathcal{B} randomly chooses a coalition $\mathbf{TA}^* = (TA_1^*, \dots, TA_n^*)$ from all possible non-empty combinations of the k TA 's. The numbering assignment of TA 's in the coalition is assigned randomly.

2. \mathcal{B} chooses a challenge pattern \mathbf{P}^* as follows. First, \mathcal{B} randomly assigns a family $t_i \xleftarrow{\$} \{0, 1, \dots, q_H\}$ for all $i \in \{1, \dots, \ell\}$. If $t_i = 0$ then \mathcal{B} assigns $P_i = *$. For $t_1 \neq 0$ \mathcal{B} randomly chooses $P_1 \xleftarrow{\$} \{TA_1, *\}$. For $i > 1$ with $t_i \neq 0$ \mathcal{B} randomly chooses $P_i \xleftarrow{\$} \mathcal{ID}$.
3. The challenger responds with WIBE parameters $param = (\hat{g}_1, \hat{g}_2, \hat{u}_{0,0}, \dots, \hat{u}_{L,1})$. \mathcal{B} initiates \mathcal{A} with the same parameters and a family of hashes H_i for each level i allowed in the HIBE.
4. Since both \mathcal{B} and \mathcal{A} are in WIBE environments playing the chosen plaintext game, they have access to the same set of oracles with \mathcal{A} having access to an additional random oracle. To prepare for random oracle queries from \mathcal{A} , \mathcal{B} initializes associative lists J_i for each level i allowed in the HIBE to answer queries for each oracle H_i . Initially empty, \mathcal{B} must either track the number of entries made on each list with a counter or equivalently through measure of its size, we let $|J_i| \leftarrow 1$ denote the size of empty lists. When \mathcal{A} queries random oracle H_i on identity ID , \mathcal{B} answers as follows:
 - If $J_i(ID)$ has been previously defined, then \mathcal{B} returns $J_i(ID)$.
 - Otherwise:
 - For $t_i = J_i$, if $i = 1$ then \mathcal{B} assigns the value $J_1(ID) \leftarrow TA_1$. If $i \neq 1$ then \mathcal{B} assigns the value $J_i \leftarrow P_i^*$.
 - For $t_i \neq J_i$, \mathcal{B} assigns $J_i(ID) \leftarrow H_i(ID)$ and increments the counter $|J_i|$.
 - \mathcal{B} then returns the value $J_i(ID)$.
5. For other queries, \mathcal{B} hashes the relevant identity pattern \mathbf{P} as a random oracle query with result \mathbf{P}' . Note that for TA -level queries with $\mathbf{ID} = (TA_i)$ the pattern is still transformed as $\mathbf{ID}' = (J_1(TA_i))$. \mathcal{A} may query:
 - **CreateTA**(TA_i) \mathcal{B} computes $TA'_i = J_1(TA_i)$ and forwards the query using its own oracle as **CreateTA**(TA'_i), returning the result.
 - **Corrupt**(\mathbf{P}) If $J(\mathbf{P}) \in_* \mathbf{P}^*$ then \mathcal{B} aborts because it would have to make an illegal query. Otherwise, \mathcal{B} computes $\mathbf{P}' = J(\mathbf{P})$ and forwards the query using its own oracle as **Corrupt**(\mathbf{P}'), returning the result.
 - **SetupCoalitionBroadcast**(TA_i, \mathcal{C}) For each TA_j in the coalition \mathcal{C} , \mathcal{B} computes the hashed TA identity $TA'_j = J_1(TA_j)$ and adds it to coalition \mathcal{C}' . \mathcal{B} also computes the hashed identity $TA'_i = J_1(TA_i)$ then forwards the query as **SetupCoalitionBroadcast**(TA'_i, \mathcal{C}'), returning the result.
 - **SetupCoalitionKeys**($TA_j, sk_j, \mathcal{C}, \hat{W}_j$) For each TA_j in the coalition \mathcal{C} , \mathcal{B} computes the hashed TA identity $TA'_j = J_1(TA_j)$ and adds it to coalition \mathcal{C}' . \mathcal{B} then computes $TA'_j = J_1(TA_j)$ and forwards the query as **SetupCoalitionKeys**(TA'_j, sk_j, \mathcal{C}), returning the result.
 - **ExtractCoalitionKey**($\mathcal{C}, v_j, d_{\mathbf{ID}}$) For each TA_j in the coalition \mathcal{C} , \mathcal{B} computes the hashed TA identity $TA'_j = J_1(TA_j)$ and adds it to coalition \mathcal{C}' . \mathcal{B} then computes $\mathbf{ID}' = J(\mathbf{ID})$ and queries **ExtractCoalitionKey**($\mathcal{C}', v_j, d_{\mathbf{ID}'}$), returning the result.
6. \mathcal{A} ends this stage of the game by outputting a challenge pattern $\hat{\mathbf{P}} = (\hat{P}_1, \dots, \hat{P}_\ell)$ and challenge coalition $\widehat{TA} = \{\widehat{TA}_1, \dots, \widehat{TA}_k\}$ and two equal-length messages (m_0, m_1) . If any $J_i(\hat{P}_i)$ has not yet been defined then \mathcal{B} sets $J_i(\hat{P}_i) \leftarrow H_i(\hat{P}_i)$.
7. \mathcal{B} must abort if:
 - if $\ell \neq \ell^*$
 - if there exists $i \in W(\hat{\mathbf{P}})$ such that $i \notin W(\mathbf{P}^*)$
 - if $J_i(\hat{P}_i) \neq P_i^*$
8. If \mathcal{B} has not aborted it outputs the messages (m_0, m_1) to the challenger.
9. The challenger computes the challenge ciphertext C^* by encrypting message m_β for $\beta \xleftarrow{\$} \{0, 1\}$ and returns it to \mathcal{B} .
10. \mathcal{B} in turn returns the challenge ciphertext to \mathcal{A} , which outputs a bit $\hat{\beta}$ as a guess for the value of β .
11. In turn, \mathcal{B} outputs $\hat{\beta}$ as its guess for the value of β .

The algorithm \mathcal{B} wins the IND-sWID-CPA game if \mathcal{A} wins the IND-WID-CPA game and \mathcal{B} does not abort. \mathcal{B} may abort if it has guess the challenge coalition and pattern incorrectly, or if it was forced to make an illegal query. \mathcal{B} uses the t_i 's to guess which of \mathcal{A} 's queries will be used to define the i^{th} level of its challenge pattern with $t_i = 0$ corresponding to a wildcard.

To avoid \mathcal{B} 's making of an illegal query we require that there are no collisions in the hash. For distinct identifiers $ID \neq ID'$ we require that the hash output $H_i(ID) \neq H_i(ID')$. With a random oracle such a collision could only occur by chance. Given $(q_H + 1)$ random oracle queries for a level the probability of a collision has an upper bound of $(q_H + 1)/|ID|$. Then the probability of successfully avoiding a collision has a lower bound of $(1 - (q_H + 1)/|ID|)$. With L independent random oracles, the probability of avoiding a collision on all levels is then $(1 - (q_H + 1)/|ID|)^L$. If a collision occurs then \mathcal{B} will gain no benefit from \mathcal{A} 's advantage as it cannot properly simulate the WIBE environment. Of \mathcal{A} 's original advantage ϵ , \mathcal{B} may only leverage an advantage of $\epsilon(1 - (q_H + 1)/|ID|)^L$.

We require that \mathcal{B} correctly choose the challenge pattern and coalition in order to win the game. The probability that $\ell = \ell^*$ is $1/L$, the probability that t_i correctly identifies P_i is $1/(q_H + 1)$ for each level $i \leq L$, and the probability of the challenge coalition \mathcal{C} being correct is $1/2^n$. Each of these factors reduce \mathcal{B} 's advantage multiplicatively in comparison to \mathcal{A} 's. If \mathcal{B} correctly guesses these values and \mathcal{A} never forces \mathcal{B} to make an illegal query then \mathcal{B} has advantage:

$$\epsilon' \geq \left(\frac{\epsilon}{2^n L (q_H + 1)^L} \right) \left(1 - \frac{(q_H + 1)}{|ID|} \right)^L$$

□

References

- [ACD⁺06] M. Abdalla, D. Catalano, A. W. Dent, J. Malone-Lee, G. Neven, and N. P. Smart. Identity-based encryption gone wild. In M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, editors, *ICALP 2006: 33rd International Colloquium on Automata, Languages and Programming, Part II*, pages 300–311. Springer-Verlag, 2006.
- [BB04] D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT '04*, pages 223–238. Springer-Verlag, 2004.
- [BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology—EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Berlin: Springer-Verlag, 2005.
- [BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *CCS '93: Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73, New York, NY, USA, 1993. ACM.