Multiple Hierarchy Wildcard Encryption

All Who Want To Play

No Institute Given

Abstract. expressionism is an art movement typically characterised by its non-realistic representation of non-tangible nouns (such as emotions or situations). It centred in the New York in the post-surrealistic decades after the second World War, and, in particular, around Peggy Guggenheim's gallery "Art of this Century".

1 Syntax

A multi-hierarchy WIBE consists of the following PPT algorithms/protocols:

- $\mathtt{Setup}(1^k, TA)$: This algorithm is run once by a TA and outputs a master public key and master private key for that TA $(mpk_{TA}, msk_{TA}) \xleftarrow{\$} \mathtt{Setup}(1^k, TA)$.

  What's to prevent an attacker setting up his own TA under the name of a real coalition member and then hijacking the update coalition protocol? Are we going to assume trusted distribution of master public keys?

- $\mathtt{SetupCoalitionBroadcast}(TA, msk_{TA}, (TA_1, mpk_{TA_1}), \ldots, (TA_n, mpk_{TA_n}))$: This algorithm creates a coalition between a set of TAs $C = (TA_1, \ldots, TA_n)$. This algorithm outputs a list of messages w_i to be sent to the TA TA_i $((TA_1, w_{TA_1}), \ldots, (TA_n, w_{TA_n}))$.
- $\mathtt{SetupCoalitionKeys}(TA, msk_{TA}, (TA_1, mpk_{TA_1}, w_1), \ldots, (TA_n, mpk_{TA_n}, w_n))$: The algorithm completes the setup of the coalition. After every member TA_i of the coalition has provided a message w_i for TA. It outputs a message u_{TA} to be broadcast to every member of its hierarchy.

The system should be able to dynamically update the coalition. We may wish to change a coalition C into a coalition C'. We assume that members $C \cap C'$ execute the `UpdateCoalition` algorithms, while new members $C \setminus C'$ execute the `JoinCoalition` algorithms. Excluded members $C' \setminus C$ are simply informed that they are no longer members of the coalition.

- $\mathtt{UpdateCoalitionBroadcast}(TA, msk_{TA}, (TA_1, mpk_{TA_1}), \ldots, (TA_n, mpk_{TA_n}))$: This algorithm updates an existing coalition C contain TA to become a new coalition $C' = (TA, TA_1, \ldots, TA_n)$. This algorithm outputs a list of messages w_i to be sent to the TA TA_i. It should be noted that some w_i may be empty, particularly if $TA_i \in C$.
- $\mathtt{Join\ CoalitionBroadcast}(TA, msk_{TA}, (TA_1, mpk_{TA_1}), \ldots, (TA_n, mpk_{TA_n}))$: A new authority TA which is joining an existing coalition to form a new coalition $C' = (TA, TA_1, \ldots, TA_n)$ uses this algorithm to produce a series of messages w_i to be sent to TA_i.
- $\mathtt{UpdateCoalitionKeys}(TA, msk_{TA}, (TA_1, mpk_{TA_1}, w_1), \ldots, (TA_n, mpk_{TA_n}, w_n))$: The algorithm completes the updating of the coalition for existing members. After every member TA_i of the coalition has provided a (non-empty) message w_i for TA. It outputs a message u_{TA} to be broadcast to every member of its hierarchy.
- $\mathtt{JoinCoalitionKeys}(TA, msk_{TA}, (TA_1, mpk_{TA_1}, w_1), \ldots, (TA_n, mpk_{TA_n}, w_n))$: This algorithm completes the joining of an existing coalition for new members. After every member TA_i of the coalition has provided a (non-empty) message w_i for TA, this algorithm outputs a message u_{TA} to be broadcast to every member of its hierarchy.

We now describe the algorithms required by the individual users.

- $\mathtt{Extract}(\boldsymbol{ID}, ID', d_{\boldsymbol{ID}})$: This algorithm outputs a decryption key $d_{\boldsymbol{ID}\|ID}$ for the identity $\boldsymbol{ID}\|ID$. The basic level has $ID = TA$ and $d_{TA} = msk_{TA}$.

- ExtractCoalitionKey$((TA_1, \ldots, TA_n), u_{TA}, d_{ID})$: This algorithm outputs a user key c_{ID} for the coalition $C = \{TA, TA_1, \ldots, TA_n\}$ by combining the broadcast key u_{TA} and their decryption key d_{ID}.
- UpdateCoalitionKey$((TA_1, \ldots, TA_n), u_{TA}, c_{ID}, d_{ID})$: This algorithm outputs an updated user key c'_{ID} for the coalition $C = \{TA, TA_1, \ldots, TA_n\}$ by combining the broadcast key u_{TA} with the user's decryption key d_{ID} and existing coalition key c_{ID}.
- Encrypt$((TA_1, mpk_{TA_1}), \ldots, (TA_1, mpk_{TA_1}), P, m)$: This algorithm is used to encrypt a message m to entities satisfying the pattern P under the coalition formed by $(TA_1, \ldots, TA_n)$. It outputs a ciphertext C or the invalid symbol $\perp$.
- Decrypt: It does what you'd expect...

2 Security Model

The security model is parameterised by a bit b involves a PPT attacker $\mathcal{A}$ which is initially given the input 1^k and access to the following oracles:

- CreateTA(TA): The oracle computes $(mpk_{TA}, msk_{TA}) \xleftarrow{\$}$ Setup$(1^k, TA)$ for the TA identity TA and returns mpk_{TA}. This oracle can only be queried once for each identity TA.
- SetupCoalitionBroadcast$(TA, (TA_1, \ldots, TA_n))$: This oracle runs the SetupCoalitionBroadcast algorithm on the appropriate inputs and returns $(w_1, \ldots, w_n)$.
- SetupCoalitionKeys$(TA, (w_1, \ldots, w_n))$: This oracle can only be queried if TA has been queried to the SetupCoalitionBroadcast oracle with n TA's in the coalition. The oracles runs the SetupCoalitionKeys algorithm assuming that message w_i was sent by TA_i. Note that this does not imply that all the TAs believe that they're in the same coalition.
- UpdateCoalition oracles are similar to the above...
- CorruptTA(TA): The oracle returns msk_{TA} and records that TA is corrupt.
- CorruptUser$(TA, \boldsymbol{ID})$: This oracle returns $d_{\boldsymbol{ID}}$ for the identity $\boldsymbol{ID}$ under the authority TA. Note that given $d_{\boldsymbol{ID}}$ the attacker can compute any coalition key $c_{\boldsymbol{ID}}$.
- UserDecrypt$(TA, \boldsymbol{ID}, C^*)$: This oracle decrypts the ciphertext with the decryption key $d_{\boldsymbol{ID}}$.
- CoalitionDecryption$(TA, \boldsymbol{ID}, C^*)$: This oracle decrypts the ciphertext with the decryption key $c_{\boldsymbol{ID}}$.
- Test$(TA_1, \ldots, TA_n, P, m_0, m_1)$: This oracle takes as input two messages (m_0, m_1) of equal length. It encrypts the message m_b for the coalition using $(mpk_{TA_1}, \ldots, mpk_{TA_n})$ under the pattern P. This oracle may only be access once and outputs a ciphertext C^*. We will let C^* denote the challenge coalition $(TA_1, \ldots, TA_n)$.

The attacker terminates by outputting a bit b'. The attacker's advantage is defined to be:

$$Adv_{\mathcal{A}}^{\texttt{IND}}(k) = |Pr[b' = 1|b = 1] - Pr[b' = 1|b = 0]| .$$

The disallowed oracle queries: (1) a CorruptTA query for any TA in the test coalition , (2) a CorruptUser query for any user ID matching the pattern P under an authority TA in the test coalition if there has been a SetupCoalitionKeys or UpdateCoalitionKeys query for the test coalition, (3) a decrypt query for C^* and any user ID matching the pattern P under an authority TA in the test coalition if there has been a SetupCoalitionKeys or UpdateCoalitionKeys query for the test coalition.