Edna: Data Disguises for Web Applications

Anonymous Author(s)

1 System Components

- Disguising Tool ("Edna"): adds support for disguise API calls to applications.
- Principal: an application user account that owns DB objects that are correlated via e.g., foreign key relationships to the account.
- Client: external devices that contact the application via the application API, and can be authenticated to speak for one or more principals via the application authentication protocol (e.g., logging in with a password, or using access tokens to establish a session).
- Tokens:
  - Correlation tokens (linking an old principal to a created principal from disguise decorrelation operations). These tokens give Edna the ability to disguise the data of these principals when disguising the data of P, or to permanently recorrelate data back with P.
  - Data tokens save the pre-modified or removed data. These tokens give Edna the ability to compose a future disguise on top of the original data, or permanently restore the original data.
  - Private key tokens save the private key of a created anonymous principal
- Disguise Capability: authorizes access to tokens.

2 Threat Model

An adversary can access any data stored as plaintext by the application or Edna, including data not exposed via its client-facing API. We assume that an adversary cannot access prior snapshots of the application database, or observe modifications or memory accesses performed during disguise application. Furthermore, undisguised content and implicit correlations based on this content (e.g., a comment that mentions the author's name) are out of scope.

We assume Edna is honest but curious: when applying or reversing a disguise, Edna may temporarily hold (in memory) plaintext token data, as well as private or symmetric keys, but is trusted to forget them once the disguise action is complete.

Finally, we assume standard security of public key and symmetric key primitives [under a random oracle model?].

3 Disguise API

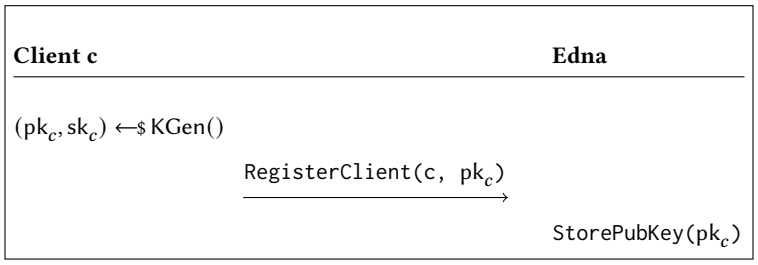

Figure 1. Client protocol for account registration

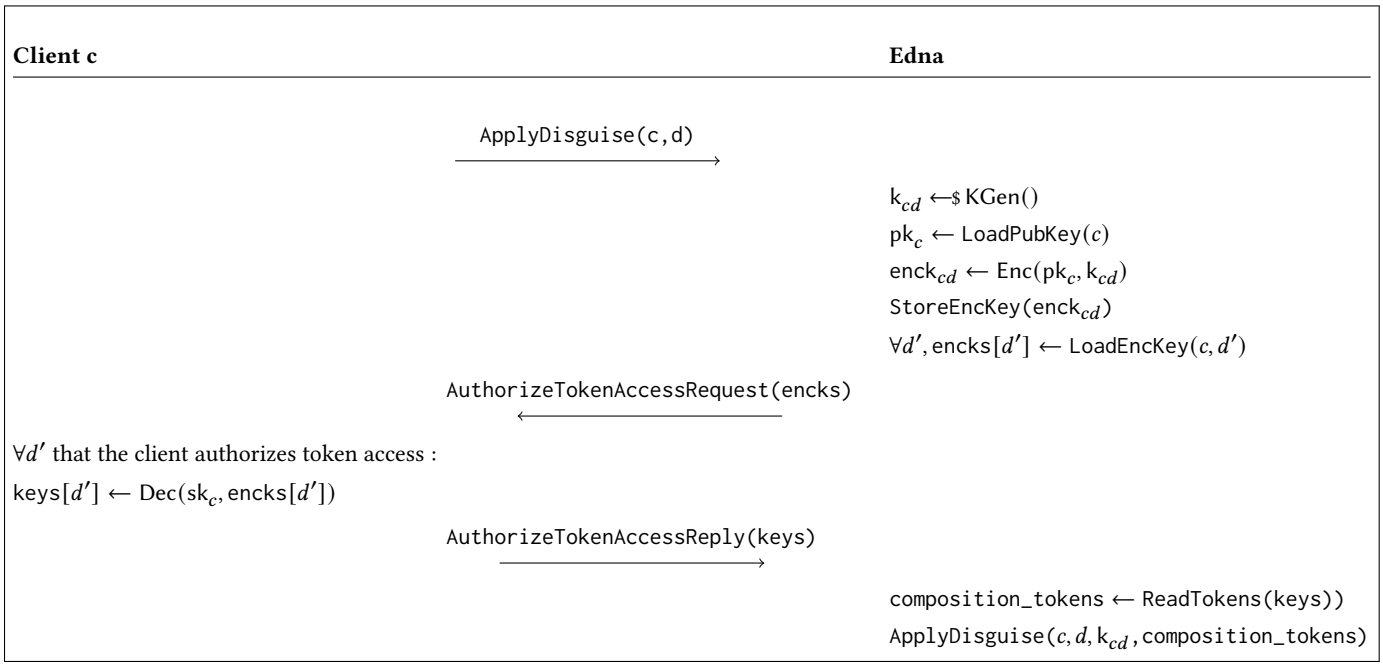

Figure 2. Client protocol for disguise application

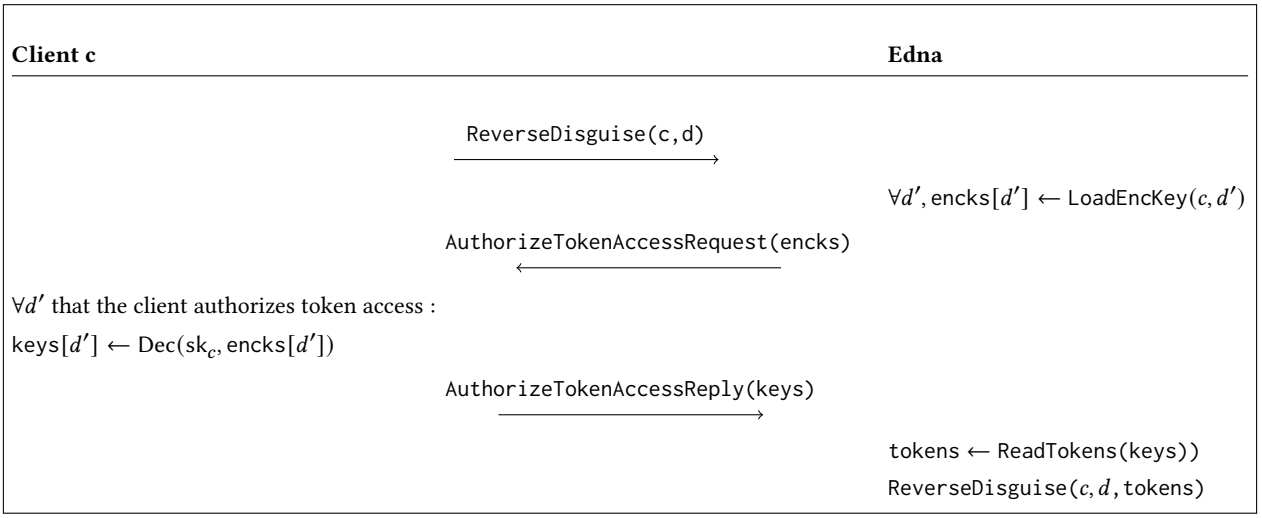

Figure 3. Client protocol for disguise reversal

```
ReadTokens(keys)

tokens ← []
for k_cd in keys:
  encToken ← LoadEncTokenListTail(c, d)
  while encToken != NULL:
    token ← Dec(k_cd, encToken)
    encToken ← token.nextEncToken
    if token.type = AnonPrivateKey:
      c' ← token.anon_uid
      sk_c' ← token.priv_key
      keys' ← []
      foreach d' :
        enck ← LoadEncKey(c', d')
        keys'.append(Dec(sk_c', enck))
        tokens.extend(ReadTokens(keys'))
      endforeach
    else :
      tokens.append(token)
    fi
  endwhile
endfor
return tokens
```

```
ApplyDisguise(c, d, k_cd,composition_tokens)

for op ∈ GetDisguise(d)
  token ← op.execute(c,composition_tokens)
  token.nextEncToken ← LoadEncTokenlistTail(c, d)
  token.nonce ← RandomU64()
  encToken ← Enc(k_cd, token)
  StoreEncTokenListTail(c, d, encToken)
endfor
```

```
ReverseDisguise(c, d,tokens)

Apply missed updates from disguise log to tokens of disguise d
Reveal data in tokens of disguise d
Undo modifications to any prior disguise's tokens by disguise d
```

Figure 4. Edna Algorithms (assuming all tokens are user-vault encrypted tokens, and ignoring global vault tokens for now)