Revocable Anonymous Credentials from Attribute-Based Encryption

Giovanni Bartolomeo[1]

[1]CNIT, Università di Roma Tor Vergata, Via del Politecnico, 1, 00133 Roma, Italy

Abstract

By leveraging on Ciphertext-Policy Attribute-Based Encryption, we build a credential management protocol with anonymous proof of predicates. The protocol supports efficient revocation through accumulators.

Keywords

Cryptography, Attribute-Based Encryption, Anonymous Credentials

1. Introduction

Anonymous credentials have experimented a renewed interest during the very last few years due to the going to mainstream of various user "wallet" models. For example, an ongoing effort at IEFT is intended to promote a very recent efficient construction of the BBS signature [1] as a standard cryptographic primitive for the problem space of privacy preserving identity credentials. Historically, anonymous credentials were mostly built on special signature schemas. The prover, after obtaining a signature over a set of attributes from an issuer, is able to randomize it and proves in zero-knowledge its possession to a verifier, optionally revealing a subset of those attributes (so called *selective disclosure*). The verifier is unable to determine which signature was used to generate the proof, removing any source of correlation (*unlinkability*).

However, in practical contexts, selective disclosure is not the only desired feature. For example, a service may require that their users are over 18 years old and that they are based in one of the European Countries. In such a case, an *anonymous proof of predicates* proving user's attribute `age` and `country` satisfying the following policy:

```
age GT 18 AND country ONEOF {Austria, Belgium,. . . , Sweden}
```

would be needed. Recent advances [2] suggest it is relatively easy to build credential systems efficiently supporting anonymous proof of predicates by functional encryption: given a ciphertext encoded using a policy, a prover can simply decrypt such a ciphertext to convince a verifier that she knows a key for a set of attributes that matches the policy. According to the authors of [2], such *functional credentials* would subsume all known credentials, such as anonymous, delegatable, or attribute-based credentials.

In this paper, we provide the following contributions:

Joint National Conference on Cybersecurity (ITASEC & SERICS 2025), February 03-8, 2025, Bologna, IT
giovanni.bartolomeo@uniroma2.it (G. Bartolomeo)

1. We directly build a functional credentials schema from the Ciphertext Policy Attribute-Based Encryption (CP-ABE) construct proposed in [3], without incurring in few extra commitment steps introduced by the author of [2] (originally required to leave their framework agnostic from any specific functional encryption schema). We consider this result important in practice, because many existing authentication protocols (such as HTTP and OAuth) use a three steps procedure (request-challenge-response), so avoiding any extra step would perfectly fit them.
2. Revocation for functional credentials is still unclear and not investigated, while it is very relevant in real world applications. We propose to augment the above schema with an efficient anonymous revocation feature leveraging on the simple dynamic accumulator originally proposed in [4].
3. We achieve performance comparable to state of the art solutions without incurring in complex zero-knowledge proof algorithms, but solely relying on a consolidated attribute-based encryption schema. Again, this result is important in practice: for example, to quickly build credential systems with anonymous proof of predicates, developers can directly rely on the well-known OpenABE framework [5] leveraging on the wide policy expressiveness this framework supports.

2. CP-WATERS-KEM and Accumulators

2.1. Preliminaries

The original construction reported in Section 5 of [3] (henceforth CP-WATERS-KEM) makes use of a bilinear group, defined as follows:

Let G and G_T be two multiplicative cyclic groups of prime order p. Let g be a generator of G and e be a bilinear map: $e\colon G \times G \to G_T$. The bilinear map e has the following properties:

1. Bilinearity: for all $u, v \in G$ and $a, b \in Z_p$, we have $e(u^a, v^b) = e(u^b, v^a) = e(u, v)^{ab}$.
2. Non-degeneracy: $e(g, g) \neq 1$.

If the group operation in G and the bilinear map e are both efficiently computable, G is said a bilinear group.

Here, we consider CP-WATERS-KEM in its small universe construction, however, an extension to the large universe construction (reported in Appendix A of [3]) is straightforward.

We implement a revocation scheme preventing decryption of ciphertext created after the key has been revoked. Note that this is not a general revocation scheme for ABE. Rather, it is a “forward revocation”, where only ciphertext generated after the actual revocation happens becomes hard to decrypt if the key has been revoked. To implement this kind of revocation, the original CP-WATERS-KEM scheme is slightly altered and combined with the cryptographic accumulator based on bilinear mappings described by Camenisch in [4].

The accumulator makes use of a set of indexes $\{i\}$ kept by the Authority and assigned to each released secret key. In the setup phase, the Authority initially creates an accumulator $acc_0 = 1$ and two public initially empty sets: $V = \{\}$ and $U = \{\}$, where U is the set of all indexes i that will be ever added to the accumulator (but may have been subsequently removed).

The sequence $g^{\gamma}, \ldots, g^{\gamma^n}, g^{\gamma^{n+2}}, \ldots, g^{\gamma^{2n}}$ (but not $g^{\gamma^{n+1}}$) is made public by the Authority (e.g., as part of MPK). Appendix D of [4] suggests a possible technique to reduce the size of this sequence. The mathematical definition of the accumulator is the following:

$$acc_V = \prod_{j \in V} g^{\gamma^{n+1-j}} = g^{\frac{\sum_{j \in V} \gamma^{n+1+i-j}}{\gamma^i}}$$

$$wit_i = \prod_{j \neq i} g^{\gamma^{n+1+i-j}} = g^{\sum_{j \neq i} \gamma^{n+1+i-j}}$$

$$\sum_{j \in V} \gamma^{n+1+i-j} = \gamma^{n+1} + \sum_{j \neq i} \gamma^{n+1+i-j} \Leftrightarrow i \in V$$

Where $g \in G$ is a generator of the group G of prime order p, and γ is picked at random from Z_p.

2.2. Revocable Functional Credentials

The schema is adapted from [2]. A revocable functional credentials scheme for an attribute universe Ω and a family of policies Φ consists of the following probabilistic algorithms:

1. $\mathtt{MSK}, \mathtt{MPK} \leftarrow \mathtt{CKGen}(1^{\lambda})$: The key generation algorithm takes input the security parameter $\lambda \in \mathtt{N}$ and outputs a key pair $(\mathtt{MSK}, \mathtt{MPK})$ of an issuer (master key pair).
2. $\mathtt{cred_i}, \mathtt{MPK}' \leftarrow \mathtt{GrantCred}(\mathtt{MSK}, \mathtt{S_i})$: The grant credential algorithm takes input the master secret key $\mathtt{MSK}$ and a non-empty set of attributes $\mathtt{S_i} \subset \Omega$ and outputs a credential $\mathtt{cred_i}$ with $\mathtt{i} \in \mathtt{N}$ for the corresponding set of attributes. It also outputs an updated master public key $\mathtt{MPK}'$.
3. $\mathtt{b} \leftarrow < \mathtt{ShowCred}(\mathtt{MPK}, \mathtt{cred_i}, \mathtt{f}), \mathtt{VrfyCred}(\mathtt{MPK}, \mathtt{f}) >$: $\mathtt{ShowCred}$ takes input the master public key $\mathtt{MPK}$, a credential $\mathtt{cred_i}$, and a policy $\mathtt{f} \in \Phi$; $\mathtt{VrfyCred}$ inputs the master public key $\mathtt{MPK}$ and a policy $\mathtt{f}$. At the end, $\mathtt{VrfyCred}$ outputs either $\mathtt{0}$ or $\mathtt{1}$.
4. $\mathtt{MPK}' \leftarrow \mathtt{Revoke}(\mathtt{MPK}, \mathtt{MSK}, \mathtt{cred_i})$: takes input the master public key $\mathtt{MPK}$, the master secret key $\mathtt{MSK}$, and a credential $\mathtt{cred_i} \in \mathtt{GrantCred}$ and outputs an update master public key $\mathtt{MPK}'$. $\mathtt{cred_i}$ is said a "revoked credential". A non-revoked credentials is said a "valid credential".

By definition, for all $\lambda \in \mathtt{N}$, for all $(\mathtt{MSK}, \mathtt{MPK}) \in \mathtt{CKGen}(1^{\lambda})$ for all $\mathtt{S} \subset \Omega$, for all $\mathtt{cred} \in \mathtt{GrantCred}$, for all $\mathtt{f} \in \Phi$ such that $\mathtt{f}(\mathtt{S}) = 1$, a functional credentials scheme

- is said *correct* if, assumed $\mathtt{cred}$ is valid (i.e., $\mathtt{cred} \notin \mathtt{Revoke}$) it holds that

$$\mathtt{Pr}[1 \leftarrow < \mathtt{ShowCred}(\mathtt{MPK}, \mathtt{cred}, \mathtt{f}), \mathtt{VrfyCred}(\mathtt{MPK}, \mathtt{f}) >] = 1$$

- is said *unforgeable* if, chosen an arbitrary policy $\mathtt{f}$, any adversary having access to all system issued credentials $\mathtt{cred_i} \in \mathtt{GrantCred}$ but the ones satisfying the policy (i.e, $\mathtt{f}(\mathtt{cred_i}) \neq 1$) and to all revoked credentials $\mathtt{cred_j} \in \mathtt{Revoke}$, has a negligible probability to succeed in the credential verification process.

- is said *anonymous* if, arbitrarily chosen a policy $\mathtt{f}$ and two provers $\mathtt{P}_0$ and $\mathtt{P}_1$ owing non-revoked credentials $\mathtt{cred_i}, \mathtt{cred_j} \notin \mathtt{Revoke}$, both satisfying or not the policy (i.e., $\mathtt{f}(\mathtt{cred_i}) = \mathtt{f}(\mathtt{cred_j})$), any adversary cannot distinguish between them.

Note that we leave out one optional feature described in the original paper (policy hiding).

2.3. Construction

To support revocable functional credentials as above defined, CP-WATERS-KEM is modified as follows:

1. $(\mathtt{MPK}, \mathtt{MSK}) \leftarrow \mathtt{Setup}(1^\lambda)$: The algorithm outputs the master secret key MSK and the master public key MPK, and publishes the MPK.
   - The algorithm chooses a group G of prime order p and generator g, random group elements $h_1, h_2, \dots, h_u$ (where u is the maximum number of system attributes) and a bilinear pairing e such that $e\colon G \times G \to G_T$. In addition, it chooses random exponents $\alpha, a, b, \gamma \in Z_p$.
   - The algorithm initially creates an accumulator $acc_0 = 1$, and two initially empty public sets: $V = \{\}$ and $U = \{\}$, where U is the set of all indexes i that will be ever added to the accumulator (but may have been subsequently removed).
   - The public key is $g, g^b, h_1, h_2, \dots, h_u, acc_V, acc_V^a$ and
   $$e(acc_V, g)^\alpha e(g^b, g^{\gamma^{n+1}})$$
   The master secret key is $g^\alpha, \alpha, a, b, \gamma$.
   - The sequence $g^\gamma, \dots, g^{\gamma^n}, g^{\gamma^{n+2}}, \dots, g^{\gamma^{2n}}$ (but not $g^{\gamma^{n+1}}$) is made public by the Authority.
2. $(\mathtt{MPK}', \mathtt{SK_i} = (\mathtt{K_i}, \mathtt{L_i}, \forall_{\mathtt{x} \in \mathtt{S}}\, \mathtt{K_{i,x}}, \mathtt{wit_i})) \leftarrow \mathtt{KeyGen}(\mathtt{MPK}, \mathtt{MSK}, \mathtt{S})$: Key generation happens by taking as input the master keys (MSK, MPK) and a set of attributes S that describe the key. The output is a randomized secret key. The Authority associates an index i to each new generated secret decryption key SK_i.
   - The algorithm includes i in the set V and U: $V = V_{old} \cup \{i\}$, $U = U_{old} \cup \{i\}$ and updates the accumulator acc_V:
   $$acc_V = \prod_{j \in V} g^{\gamma^{n+1-j}} = g^{\frac{\sum_{j \in V} \gamma^{n+1+i-j}}{\gamma^i}}$$
   - The algorithm updates the terms in the master public key using the accumulator: acc_V^a and $e(acc_V, g)^\alpha e(g^b, g^{\gamma^{n+1}})$, while the master secret key MSK remains the same.
   - Chosen a random $t \in Z_p$, the algorithm computes the secret decryption key component K_i as as follows: $K_i = g^{\alpha + abt + b\gamma^i}$, $L_i = g^{bt}$ and, for each $x \in S$, $K_{i,x} = h_x^t$.
   - Also, the algorithm releases a new key component (the witness):
   $$wit_i = \prod_{j \neq i} g^{\gamma^{n+1+i-j}} = g^{\sum_{j \neq i} \gamma^{n+1+i-j}}.$$

3. $\mathtt{MPK'} \leftarrow \mathtt{KeyRemove(PK, MSK, i)}$: A similar step is also executed when the Authority needs to revoke a key K_i. In this case, the algorithm simply removes i from the set V, recomputes the accumulator value acc_V and, consequently, the terms in the master public key using it: acc_V^a and $e(acc_V, g)^{\alpha} e(g^b, g^{\gamma^{n+1}})$ as above (the master secret key MSK remains the same).
With the addition or removal of elements to the accumulator, previously released witnesses become stale and any Client who has previously received a witness wit_i shall update it. Therefore, the following step is introduced into the schema:
4. $\mathtt{wit'_i} \leftarrow \mathtt{UpdateWitness(MPK, V_{old}, V, wit_i)}$: The algorithm takes as input the old witness and updates it to match the new master public key MPK and the current set of authorized indices V. The new witness is locally computed using the following equation:

$$wit'_i \leftarrow wit_i \frac{\prod_{j \in V/V_{old}} g^{\gamma^{n+1+i-j}}}{\prod_{j \in V_{old}/V} g^{\gamma^{n+1+i-j}}}$$

Note that the Client does not know $g^{\gamma^{n+1}}$, hence, this algorithm fails when the condition $i \in V \cap V_{old}$ is not verified, i.e. a Client cannot update its wit_i if i is not (no more) in V as a result of a revocation. In this case the update operation returns $wit'_i = \perp$.
5. $(\mathtt{C} = (\mathtt{C'}, \mathtt{C''}, \forall_{\mathtt{k} \in [1, \ldots \mathtt{l}]}\ \mathtt{C_k}), \mu) \leftarrow \mathtt{Encrypt(MPK, M^{l \times m}}, \rho)$: The algorithm takes as input an access structure (M, ρ) and the public key MPK. M is an $l \times n$ matrix, while ρ is an injective function associating each row of M to an attribute ρ_k (i.e.,$\rho_k = \rho(k) \in S$); note that in this construct one attribute is associated with at most one row. The output is a random secret and the ciphertext.
   - Chosen a random vector $\vec{v} = (s, y_2, \ldots, y_m)$ in Z_p^n and being M_k the k-th row of M, the algorithm computes $\lambda_k = v \cdot M_k$.
   - Together with a with description of (M, ρ), the algorithm makes public the ciphertext:
   $$C' = g^{bs}, C_k = acc_V^{a\lambda_k} h_{\rho_k}^{-s}, C'' = acc_V^s = (\prod_{j \in V} g^{\gamma^{n+1-j}})^s$$
   - Finally, the algorithm computes the random secret
   $$\mu = [e(acc_V, g)^{\alpha} e(g^b, g^{\gamma^{n+1}})]^s$$
   and keeps it private.
6. $\mu \leftarrow \mathtt{Decrypt(SK_i, C)}$: Dually, the decryption takes as input the ciphertext C and the secret key SK. The output is the shared secret if and only if the set of attributes S satisfies the access structure, or $\perp$ otherwise.
   - For each k such that $\rho_k \in S$ (i.e., consider only attributes in S), compute ω_k such that $\sum_k \omega_k \lambda_k = s$ (there could different sets of $\{\omega_k\}$ satisfying this equation)
   - Compute the random secret:
   $$\mu = \frac{e(C'', K_i)}{\prod_k [e(C_k, L_i) e(C', K_{i,\rho_k})]^{\omega_k} e(C', wit_i)} =$$
   $$[e(acc_V, g)^{\alpha} e(g^b, g^{\gamma^{n+1}})]^s$$

2.4. Correctness and Security

To understand why decryption works, consider the solution equation, and note the numerator is

$$e(C'', K) = e(g,g)^{\frac{\sum_{j\in V}\gamma^{n+1+i-j}}{\gamma^i} s(\alpha+abt+b\gamma^i)} =$$

$$e(g,g)^{\frac{\sum_{j\in V}\gamma^{n+1+i-j}}{\gamma^i}\alpha s} e(g,g)^{\frac{\sum_{j\in V}\gamma^{n+1+i-j}}{\gamma^i} sabt} e(g,g)^{bs\sum_{j\in V}\gamma^{n+1+i-j}} =$$

$$e(acc_V, g)^{\alpha s} e(acc_V, g)^{sabt} e(g^s, g)^{b\sum_{j\in V}\gamma^{n+1+i-j}}$$

As in the original CP-ABKEM decryption algorithm, the second factor $e(acc_V, g)^{sabt}$ cancels out with the first part of the denominator:

$$\prod_k [e(C_k, L)e(C', K_{\rho_k})]^{\omega_k} = \prod_k e([acc_V^{a\lambda_k} h_{\rho_k}^{-s}, g^{bt})e(g^{bs}, h_{\rho_k}^t)]^{\omega_k} =$$

$$\prod_k e(acc_V, g)^{ab\lambda_k\omega_k t} = e(acc_V, g)^{sabt}$$

Regarding the third factor $e(g^s, g)^{b\sum_{j\in V}\gamma^{n+1+i-j}}$ we note that, if and only if the index is contained in the current accumulator (i.e., $i \in V$), we have:

$$e(g^s, g)^{b\sum_{j\in V}\gamma^{n+1+i-j}} = e(g^{bs}, g^{\gamma^{n+1}+\sum_{j\neq i}\gamma^{n+1+i-j}}) =$$

$$e(g^{bs}, g^{\gamma^{n+1}})e(g^{bs}, wit_i)$$

Partially cancelling out with the factor $e(g^{bs}, wit_i)$ in the denominator. Therefore, the result of the computation is

$$[e(acc_V, g)^{\alpha} e(g^b, g^{\gamma^{n+1}})]^s$$

To prove security, we use a security game based on the one presented in Section 5 of [3]. The adversary chooses to be challenged on an encryption to an access structure A^*, and can ask arbitrarily q times for any private key S that does not satisfy A^*. However, the original model is extended by letting the adversary query for private keys that satisfy the access structure, with the restriction that any of those keys shall be revoked before the challenge:

- *Setup.* The challenger runs `Setup` algorithm and gives the public parameters, `PK` to the adversary.
- *Phase 1.* The adversary makes repeated private keys corresponding to sets of attributes $\mathtt{S}_1, \ldots, \mathtt{S}_{\mathtt{q}'}$ (with $1 < q' < q$).
- *Revocation* Using the `keyremove` algorithm in the schema, any key may (or not) be revoked. *Phase 1* and *Revocation* may be arbitrarily interleaved.
- *Challenge.* The adversary submits two equal length messages $\mathtt{M}_0$ and $\mathtt{M}_1$. In addition the adversary gives a challenge access structure $\mathtt{A}^*$ such that none of the sets $\mathtt{S}_1, \ldots, \mathtt{S}_{\mathtt{q}'}$ from *Phase 1* satisfies the access structure, or such that any of those keys corresponding to sets satisfying the access structure has been revoked. The challenger flips a random coin β, and encrypts $\mathtt{M}_\beta$ under $\mathtt{A}^*$. The ciphertext $\mathtt{CT}^*$ is given to the adversary.

- *Phase 2*. *Phase 1* is repeated with the restriction that none of sets of attributes $\mathtt{S}_{\mathtt{q}'+1}, \ldots, \mathtt{S}_{\mathtt{q}}$ satisfies the access structure corresponding to the challenge. Revocation may also occur in this phase.
- *Guess*. The adversary outputs a guess β' of β.

The advantage of the adversary in the above game is $\epsilon = \mathtt{Pr}[\beta' = \beta] - \frac{1}{2}$ and, by definition, the scheme is secure if all polynomial time adversaries have at most a negligible advantage. We use a selective proof, therefore the above game is augmented by an initial step *Init* in which the adversary commits to the challenge access structure A^* and to the final set of credentials that will eventually appear in the accumulator V^*.

Our security proof works under the General Diffie-Hellman Exponent Problem introduced by Boneh, Boyen and Goh in [6]. Using this hardness assumption, in a longer version of this paper [7] we prove that chosen an access structure A^*, no polynomial time adversary can (selectively) break our system, provided all keys satisfying A^* have been revoked before the challenge.

Note that the presented security model supports only chosen-plaintext attacks. The model is extended to handle chosen-ciphertext attacks by allowing for decryption queries in Phase 1 and Phase 2. To achieve chosen-ciphertext security we use the Fujisaki-Okamoto transformation reported in subsection 3.1 and 3.2. As this transformation exactly applies as in the original paper, we let the reader refer to [5] for the security proof.

2.5. Verification Protocol

We use the above CP-ABE schema to implement the revocable functional credential schema in section 2.2(the protocol is also adapted from [8]):

1. CP-ABE $\mathtt{Setup}(1^\lambda)$ algorithm takes place in order to generate the key pair (MPK, MSK).
2. The grant credential algorithm is implemented through the $\mathtt{MPK}', \mathtt{SK_i} \leftarrow \mathtt{KeyGen}(\mathtt{MPK}, \mathtt{MSK}, \mathtt{S})$ algorithm which releases credentials $cred_i = SK_i$ corresponding to a non-empty set of attributes S. It also outputs an updated master public key MPK'.
3. To check credential, chosen an access policy (i.e., a matrix $M^{l \times m}$, and a function ρ), a verifier generates and encrypts a random secret μ through the CP-ABE $\mathtt{Encrypt}(\mathtt{MPK}, \mathtt{M}^{\mathtt{l}\times\mathtt{m}}, \rho)$; and sends the resulting ciphertext C to the prover. Using a credential SK', the prover executes $\mu' \leftarrow \mathtt{Decrypt}(\mathtt{SK}', \mathtt{C})$ and sent back the result to the verifier. The verifier output 1 if $\mu = \mu'$ and 0 otherwise.
4. Revocation of credential $cred_i$ is implemented through the $\mathtt{KeyRemove}(\mathtt{PK}, \mathtt{MSK}, \mathtt{i})$ algorithm, which updates the master public key to MPK'.

Note that since each challenge encapsulates a randomly generated secret token, the protocol is natively immune to replay attacks.

3. Anonymity

To ensure anonymity, the Fujisaki Okamoto transformation may be applied to CP-WATERS-KEM. This transformation was already described in [5] and there proved to make the schema secure under Chosen Ciphertext Attacks (CCA). The transformation uses two random numbers r_c and K_c both chosen by the encrypting party to generate randomness for encryption. It is possible to prove that there is a negligible probability for an attacker to produce a ciphertext that may decrypt, and that this probability is the same as guessing a ciphertext without any knowledge of the randomness used to produce it. Using this property, in Appendix A we prove correctness, unforgeability and anonymity of the proposed scheme.

3.1. CCA-secure Encryption Algorithm

The CCA-secure encryption algorithm is specified by the following steps:

- The decrypting party (prover) shall choose a random number r_c and send it to the encrypting party.
- Received r_c, the encrypting party (verifier) chooses an access structure AP and a secret K_c and concatenates them to form the string $r_c||K_c||AP$
- The encrypting party runs the encryption algorithm of the original CP-WATERS-KEM or of the modified schema with revocation to get a random secret and the ciphertext. The seed $r_c||K_c||AP$ is used as a source of randomness for the encryption algorithm with $\mathtt{u} \leftarrow \mathtt{PRG}(\mathtt{H}'(\mathtt{r_c}||\mathtt{K_c}||\mathtt{AP}), \lambda)$, where PRG is a pseudo random generator, λ is the length of the returned random bit string ($u \in \{0,1\}^l$) and H' is a collision-resistant hash function.
  - The random secret is $e(g,g)^{\alpha s}$ for CP-WATERS-KEM or $(e(acc_V, g)^{\alpha} e(g^b, g^{\gamma^{n+1}}))^s$ for the modified CP-WATERS-KEM. The encrypting party keeps it private and uses in the next step.
  - The encrypting party releases the ABKEM ciphertext C_{ABKEM}.
- The encrypting party uses random secret above for XORing the concatenation $r_c||K_c$
  - Transform $r_c||K_c$ into bytes (octects).
  - Using the pseudo random generator PRG, get

    $$\mathtt{r} \leftarrow \mathtt{PRG}(\mathtt{H}(\mathtt{e}(\mathtt{g},\mathtt{g})^{\alpha \mathtt{s}}), \lambda)$$

    for CP-WATERS-KEM or

    $$\mathtt{r} \leftarrow \mathtt{PRG}(\mathtt{H}([\mathtt{e}(\mathtt{acc_V},\mathtt{g})^{\alpha}\mathtt{e}(\mathtt{g^b},\mathtt{g}^{\gamma^{n+1}})]^{\mathtt{s}}), \lambda)$$

    for the modified CP-WATERS-KEM, with H being a collision-resistant hash function.
  - Finally, compute $C = r \oplus (K_c||r_c)$

3.2. CCA-secure Decryption Algorithm

The CCA-secure decryption algorithm is specified by the following steps:

- Run decryption of the original CP-WATERS-KEM or of the modified schema to decrypt the ciphertext and obtain the shared secret:

  $$e(g,g)^{\alpha s}$$

for CP-WATERS-KEM or

$$(e(acc_V, g)^{\alpha} e(g^b, g^{\gamma^{n+1}}))^s$$

for the modified schema.

- Use that shared secret to generate randomness

$$\mathtt{r} \leftarrow \mathtt{PRG}(\mathtt{H}(\mathtt{e}(\mathtt{g},\mathtt{g})^{\alpha\mathtt{s}}), \lambda)$$

or

$$\mathtt{r} \leftarrow \mathtt{PRG}(\mathtt{H}([\mathtt{e}(\mathtt{acc_V},\mathtt{g})^{\alpha}\mathtt{e}(\mathtt{g^b},\mathtt{g}^{\gamma^{\mathtt{n+1}}})]^{\mathtt{s}}), \lambda)$$

- Use generated randomness r for XORing the ciphertext and retrieve K_c and r_c: $C \oplus r = (K_c||r_c)$
- Verify r_c matches the random number chosen at beginning.
- Run again the CCA-secure Encryption using $r_c||K_c||AP$ as a source of randomness and verify the result is equal to the received ciphertext C_{ABKEM}.

4. Related Works

Due to space limitations, we let the reader refer to [9] for a survey on revocation strategies for anonymous credentials. Furthermore, [10] reports several works considering the application of dynamic universal accumulators to anonymous credentials to implement blacklists. While these approaches require to prove in zero-knowledge that a prover's non-membership witness satisfies the accumulator verification equation, the authors describe a different construction where both the accumulator and the anonymous credentials, previously described in [11], rely on the same construct (structure-preserving signatures on equivalence classes). To same extent, our approach is similar to their one, but we highlight the different scope as [10] limits to consider *selective disclosure*, not proof of predicates. In terms of performance, scheme 2 in [11], using primitives `VerifyR` and `VerifySubset`, requires a total of $2 * (i + 2)$ pairing operations per number of credential entries i, plus two additional revocation-induced pairings (scheme 2 in [10]). Our scheme presented in section 2.3, using the optimization described in [5], reduces the number of pairings to $k + 2$, with k being the number of attributes satisfying the policy, plus one more for checking the witness.

5. Conclusion

Combining Ciphertext Policy Attribute-Based Encryption (CP-ABE) and accumulators we build a revocable credential management framework supporting anonymous proof of predicates over attributes; we further achieve anonymity by applying a simple transformation to the resulting schema. To the best of our knowledge, our work is the first one efficiently combining rich policy expressiveness (from ABE), revocation (from accumulator) and anonymous proof of predicates over attributes into a single framework.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] S. Tessaro, C. Zhu, Revisiting bbs signatures, Cryptology ePrint Archive, Paper 2023/275, 2023. URL: https://eprint.iacr.org/2023/275, https://eprint.iacr.org/2023/275.

[2] D. Deuber, M. Maffei, G. Malavolta, M. R. D. Schröder, M. Simkin, Functional credentials, Proceedings on Privacy Enhancing Technologies 2018 (2018). doi:10.1515/popets-2018-0013.

[3] B. Waters, Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization, Cryptology ePrint Archive, Paper 2008/290, 2008. URL: https://eprint.iacr.org/2008/290, https://eprint.iacr.org/2008/290.

[4] J. Camenisch, M. Kohlweiss, C. Soriente, An accumulator based on bilinear maps and efficient revocation for anonymous credentials, Cryptology ePrint Archive, Paper 2008/539, 2008. URL: https://eprint.iacr.org/2008/539, https://eprint.iacr.org/2008/539.

[5] B. Waters, M. Green, The OpenABE Design Document, Technical Report, Zeutro LLC Encryption and Data Security, 2018. URL: https://github.com/zeutro/openabe/blob/master/docs/libopenabe-v1.0.0-design-doc.pdf.

[6] D. Boneh, X. Boyen, E.-J. Goh, Hierarchical identity based encryption with constant size ciphertext, Cryptology ePrint Archive, Paper 2005/015, 2005. URL: https://eprint.iacr.org/2005/015, https://eprint.iacr.org/2005/015.

[7] G. Bartolomeo, A zero-knowledge revocable credential verification protocol using attribute-based encryption, 2024. URL: https://arxiv.org/abs/2308.06797. arXiv:2308.06797.

[8] ETSI, CYBER; Attribute Based Encryption for Attribute Based Access Control, ETSI Technical Specification TS 103 532, European Telecommunications Standards Institute, Sophia Antipolis, France, 2021.

[9] J. Lapon, M. Kohlweiss, B. De Decker, V. Naessens, Analysis of revocation strategies for anonymous idemix credentials, in: B. De Decker, J. Lapon, V. Naessens, A. Uhl (Eds.), Communications and Multimedia Security, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 3–17.

[10] D. Derler, C. Hanser, D. Slamanig, A new approach to efficient revocable attribute-based anonymous credentials, in: J. Groth (Ed.), Cryptography and Coding, Springer International Publishing, Cham, 2015, pp. 57–74.

[11] C. Hanser, D. Slamanig, Structure-preserving signatures on equivalence classes and their application to anonymous credentials, in: P. Sarkar, T. Iwata (Eds.), Advances in Cryptology – ASIACRYPT 2014, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 491–511.

A. Security

We prove the following:

Theorem. A polynomial time adversary, acting as a Verifier, cannot distinguish between any two provers with different CP-WATERS-KEM (plus Accumulator) keys, if their (non revoked) keys both satisfy (or not satisfy) the same access structure they are tested against.

Proof. We start considering the following security game (adapted from [2]):

1. The $\texttt{Setup}(1^\lambda)$ algorithm of CP-WATERS-KEM or the modified schema takes place. The public key PK is given to the adversary.
2. Any Prover P_i receives distinct secret keys K_i embedding some attributes.
3. The adversary is allowed to submit queries in the form $(r_c||K_c||AP)$ to an oracle $\texttt{O}^1$ which produces a random output u if this is the first time the input has been queried on. Otherwise, it gives back the previous response. In addition, the oracle computes the ciphertext C using the CCA-secure encryption algorithm and records the couple $((r_c||K_c||AP),(C,u))$ in a table. This oracle operation runs throughout the whole game.
4. The adversary, acting as a Verifier V, arbitrarily chooses an access structure A^* and two Provers P_0 and P_1, such that their corresponding keys either both satisfy, or both not satisfy the chosen access structure.
5. Depending on an internal coin toss b, a second oracle $\texttt{O}^2$ impersonates the prover P_b in the verification algorithm.
6. Verifier V computes a CCA-secure ciphertext and sends it to $\texttt{O}^2$.
7. $\texttt{O}^2$ responds with the decrypted ciphertext m or with $\perp$.
8. The aforementioned steps (except the Setup) are repeated adaptively for any polynomial number of times on arbitrarily chosen access structure and arbitrarily chosen pairs of provers.
9. The adversary tries a guess b' and wins the game if $b == b'$ (i.e., she is able to guess which Prover has responded).

Modify the game as follows: at step 7, when given a ciphertext C, oracle $\texttt{O}^1$ checks if C appears in the random oracle table. If so, it outputs the corresponding $m = (K_c||r_c)$ value in the table; otherwise, it outputs $\perp$ and rejects.

The difference between the original game and the modified one is negligible, as in the original game the oracle may decrypt even in case of a forged ciphertext (i.e., a ciphertext not computed using the CCA-secure encryption algorithm). However, since $\texttt{O}^1$ was not queries on $(r_c||K_c||AP)$, the probability that this event happens is bounded by the probability of apriori guessing a ciphertext output by an encryption for a given message without knowing the randomness used to encrypt.

Now, the following observations apply to this modified game:

- If the Verifier produces a genuine ciphertext C following the CCA-secure Encryption algorithm, she gets a correct decryption m if the attributes embedded in the secret key K_b satisfy the chosen access structure A^*, i.e. $A^*(K_b) = 1$. Thus, the presented schema satisfies by definition the correctness property.

- Viceversa, if the attributes embedded in the secret key K_b do not satisfy the chosen access structure A^*, i.e. $A^*(K_b) = 0$, the ciphertext wouldn't decrypt at all except for a negligible probability ϵ. Thus, the presented schema satisfies by definition the unforgeability property.

Furthermore, we observe that:

- The access structure A^* associated to the ciphertext C is always known to the challenger (given as input after being chosen by the adversary)
- Because a pseudo random generator is used, the ciphertext C is deterministically computed from the public key PK and the access structure A^*
- The ciphertext C is uniformly distributed on the ciphertext space, because computed using the uniformly distributed randomness u in step 3.

Under the conditions above, suppose to modify the previous game replacing prover P_b's behaviour as follows:

- if key K_b embeds attributes satisfying the access structure A^*, then message m is returned;
- otherwise $\perp$ is returned.

That is, P_b no longer evaluates the decryption using the key K_b rather it (deterministically) returns m or $\perp$ depending on the internal bit $A^*(K_b)$. Since $A^*(K_0) = A^*(K_1)$ (both keys satisfy or not satisfy the access structure), in the latter schema the random coin b of the oracle remains hidden in the information-theoretic sense. This implies that the advantage of any adversary is $1/2$ in distinguish between P_0 and P_1. As the introduced modifications do not alter the advantage except for at most a negligible probability, the advantage of any adversary in the original game is negligibly close to $1/2$. $\square$