Formal Analysis of a TTP-Free Blacklistable Anonymous Credentials System

Weijin Wang[1,2(✉)], Jingbin Liu[1,2], Yu Qin[1], and Dengguo Feng[1,3]

[1] TCA, Institute of Software, Chinese Academy of Sciences, Beijing, China
{wangweijin,liujingbin,qin_yu,feng}@tca.iscas.ac.cn
[2] University of Chinese Academy of Sciences, Beijing, China
[3] SKLCS, Institute of Software, Chinese Academy of Sciences, Beijing, China

Abstract. This paper firstly introduces a novel security definition for BLAC-like schemes (BLAC represents TTP-free BLacklistable Anonymous Credentials) in symbolic model using applied pi calculus, which is suitable for automated reasoning via formal analysis tools. We model the definitions of some common security properties: authenticity, non-framebility, mis-authentication resistance and privacy (anonymity and unlinkability). The case study of these security definitions is demonstrated by modelling and analyzing BLACR (BLAC with Reputation) system. We verify these security properties by Blanchet's ProVerif and a ZKP (Zero-Knowledge Proof) compiler developed by Backes *et al.*. In particular, we analyze the express-lane authentication in BLACR. The analysis discovers a known attack that can be carried out by any potential user to escape from being revoked as he wishes. We provide a revised variant that can be proved successfully by ProVerif, which also indicates that the fix provided by ExBLACR (Extending BLACR) is incorrect.

Keywords: Formal analysis · Anonymous credential · ProVerif
BLACR

1 Introduction

Anonymous credentials allow users to obtain credentials on their identities and prove possession of these credentials anonymously. There are three parties in the anonymous credentials system: *users* obtain credentials from *issuers* (or GM, indicating Group Manager). They can then present these credentials to *verifiers* (or SP, indicating Service Provider) in an anonymous manner. The verifiers can check the validity of users' anonymous credentials but cannot identify them.

Practical solutions for anonymous credentials have been proposed, such as IBM's identity mixer [19] and TCG (Trusted Computing Group)'s DAA (Direct Anonymous Attestation) protocol [14,15], Microsoft's U-Prove [22], or Nymble system [21]. To avoid misbehavior, most of schemes introduce a TTP (Trust Third Party) to revoke misbehaved users. However, having a TTP capable of

S. Qing et al. (Eds.): ICICS 2017, LNCS 10631, pp. 3–16, 2018.
https://doi.org/10.1007/978-3-319-89500-0_1

deanonymizing or linking users' access may be dangerous. Recognizing this, elimination of such TTP while still supporting revocation is desired. In this spirit, many schemes had been proposed, such as EPID [16], BLAC [27], BLACR [8], ExBLACR [29], PEREA [6], PERM [7], PE(AR)^2 [32], FARB [30]. In these schemes, SP can punish users without the assistance of TTP, and the users must convince SP that they satisfy the predetermined authentication policy in a zero-knowledge way.

All these schemes are claimed to be provable secure except for U-Prove. However, computational security definitions of these schemes are very complex, thus making the proof of security error-prone. For example, Camenisch *et al.* [18] pointed out that the known security models of DAA are non-comprehensive and even insecure recently, and gave a security model under universally composable framework. BLACR system is also reported that a feasible attack exists [29]. Recognizing this, we tend to prove these complex schemes in another perspective, namely formal methods, which are widely used to verify cryptographic protocols. We think formal analysis can help us to find the logical errors of protocols and become a complement of the computational security proof.

Fortunately, formal analysis has shown its power to prove the complex security definitions although the formal analysis of anonymous credentials is relatively limited (almost for DAA). Arapinis *et al.* [4] presented a framework for analyzing the unlinkability and anonymity in the applied pi calculus. Arapinis *et al.* [2,3] make use of this framework to analyze the privacy of composing protocols using ProVerif [13]. Smyth *et al.* [25,26] introduced a definition of privacy for DAA schemes that was suited to automated reasoning by ProVerif. They discovered a vulnerability in the RSA-based DAA scheme and fixed it to meet their definition of privacy. Xi *et al.* [31] utilized ProVerif to analyze the DAA scheme in TPM 2.0. They put forward a definition of forward anonymity for DAA scheme in symbolic model. To deal with the complex zero-knowledge proof within equational theory, Backes *et al.* [10] presented an abstraction of zero-knowledge proof that is formalized by the applied pi calculus and developed a compiler to encode this abstraction into a rewriting system that is suited to ProVerif. They also performed an analysis for DAA using this approach and found a novel attack.

Contributions. In this paper, we present a novel definition for some common security properties of BLAC-like schemes via applied pi calculus. Specifically, we formalize authenticity, non-frameability and mis-authentication resistance as correspondence properties and privacy as equivalence properties using applied pi calculus (Sect. 2).

For a case study, we analyze BLACR system (Sect. 3). We model its subprotocols by applied pi processes and defined some main processes to analyze those security properties. Our analysis result shows that the BLACR holds these security properties in the normal-lane form (Sect. 3.4). Specially, we also model and analyze the express-lane authentication of BLACR (Sect. 3.5). This analysis shows an anticipative action if a user always does not trigger the revocation conditions but reports a known vulnerability when a user have potential to get

revoked. This attack allows a user to escape from being revoked as he wishes after he owns a express-lane token, which disables the security policy of BLACR. Then we provide a revised variant that can be proved by ProVerif. The revision also shows that the fix provided by ExBLACR is incorrect.

2 Syntax and Security Definition

We adopt the process calculus of ProVerif [1,11,12,20], which is inspired by applied pi calculus [23] to define the security properties. Without ambiguity, we sometimes call it applied pi calculus instead of ProVerif calculus. Due to space limitation, the review of ProVerif calculus will be presented in the full version.

2.1 Syntax

Roughly speaking, a TTP-free blacklistable anonymous credentials system contains the following algorithms:

Initialization. This algorithm initializes the system parameters. The issuer constructs a signing key pair (pk_I, sk_I). If SP is not the issuer, then SP will construct its own key pair (pk_V, sk_V). Especially, the implementation-specific parameters will be defined, such as initializing the blacklist.

Registration. This algorithm is registration phase between the issuer and a legitimate user to enroll the user as a member in the group of registered users. Upon successful completion of this phase, The user obtains a credential signature cre on his secret value x.

Authentication. The user will generate a zero-knowledge proof to convince an SP that he has the right to obtain service. First, the user in possession of x proves that he holds a valid credential cre. Then the user convinces that he satisfies the authentication policy. Note that a *protocol transcript* τ (a ticket) must be seen by the SP to guarantee freshness and to block the authenticating user if necessary.

Verification. SP will check the validity of the received zero-knowledge proofs. If failed, the user will be blocked to access.

List Management. SP can manipulate the list with the transcript τ according to a specific authentication policy. In a reputation-based policy, the SP scores the user's action of the session with a transcript τ and executes the operation $add(L, (\tau, s))$ to add the score s to the current blacklist L.

2.2 Security Definition

In this section, we present the definitions of security properties in the symbolic model using applied pi calculus.

Assumptions and Notations. In this paper, we denote registration process as `Register` (for users) and `Issue` (for the issuer), and authentication process as `Authenticate`. Verification and list management processes can be combined together since they are all handled by SP, which is denoted as `Verify`. Initialization process will be encoded into the main process.

In process `Register`, event *registered* will be executed after the user successfully registers with the issuer and obtains a valid credential, otherwise, event *unregistered* will be executed. In process `Authenticate`, event *startAuth* represents a new authentication activated by the user. Event *acceptAuth* will be executed when the verification of zero-knowledge proofs succeeds in process `Verify`, and conversely, event *revoke* will be executed.

We assume that the adversary controls the execution of an arbitrary number of users in an arbitrary fashion except for learning their secret, as shown below:

$$\texttt{ControlUsers} = !pub(id).!vp.(\texttt{Register} \mid (p(cre).!(\texttt{Authenticate} \mid \texttt{Judge}))).$$

The adversary can choose any user (id) to run the processes `Register` and `Authenticate`. The restricted channel name p is used for delivering the credential of the user between registration and authentication.

Process `Judge` models the judgment of a user's state (for example, his current reputation score). We record two events in process `Judge`: event *satisfyPolicy* for a satisfied judgment; and event *notSatisfy* for a failure.

Authenticity. In a system with *authenticity*, an SP is assured to accept authentication only from users who satisfy the authentication policy. This definition can be parsed as the following statements:

1. SP accepts authentication from users who satisfy the authentication policy.
2. SP would never accept authentication from users who violate the policy.

Build on this understanding, we formalize *authenticity* as two correspondence properties using the events recorded in the processes.

Definition 1 (Authenticity). Given processes ⟨`Register, Issue, Authenticate, Verify, Judge`⟩, we say authenticity is satisfied if the following correspondences are held:

$$event : \mathit{acceptAuth} \rightsquigarrow \mathit{startAuth} \,\&\, \mathit{satisfyPolicy}\, is\, true.$$
$$event : \mathit{acceptAuth} \rightsquigarrow \mathit{startAuth} \,\&\, \mathit{notSatisfy}\, is\, false.$$

The correspondence event:*acceptAuth* $\rightsquigarrow$ *startAuth* & *satisfyPolicy* means that if the SP passes the verification and accepts the authentication from a user, then this user has started an authentication session and satisfied the authentication policy before. This means that the SP accepts the authentication from a user who has satisfied the policy, which is immediately corresponding to the statement 1. Similarly, the second failed correspondence is corresponding to statement 2.

Non-frameability. A user is framed if he satisfies the authentication policy, but is unable to successfully authenticate himself to an honest SP [8]. Hence, if a system satisfies *non-frameability*, then the situation that a user fails to authenticate to an honest SP but satisfies the policy should never happen.

Definition 2 (Non-frameability). Given processes in Definition 1, if correspondence event:*revoke* $\rightsquigarrow$ *startAuth* & *satisfyPolicy* is false, non-framebility is satisfied.

This correspondence means that, if SP rejects an authentication, then the situation that a user has started this authentication session and satisfied the authentication policy would never happen, which is corresponding to the statement of non-frameability.

Mis-authentication Resistance. Mis-authentication takes place when an unregistered user successfully authenticates himself to an SP. In a system with *mis-authentication resistance*, an SP is assured to accept authentications only from registered users. we can parse this description into two statements.

1. The statement "a user successfully authenticates to an SP, but he never registered to the issuer before" is false.
2. The statement "if an SP accepts the authentication from a user, then before that, this user has registered with the issuer" is true.

Naturally, we formalize the statements into the following properties.

Definition 3 (Mis-authentication Resistance). If processes in Definition 1 are given, Mis-authentication resistance is satisfied when the following correspondences are held:

$$event : \mathit{acceptAuth} \rightsquigarrow \mathit{startAuth} \,\&\, \mathit{unregistered}\, is\, false.$$
$$event : \mathit{acceptAuth} \rightsquigarrow \mathit{startAuth} \,\&\, \mathit{registered}\, is\, true.$$

Privacy. The definition of privacy is twofold: *anonymity* and *unlinkability*, which is inspired by the formal definitions in [4].

Anonymity ensures that an adversary cannot see the difference between a system in which the user with a publicly known identity id_0 executes the analyzed processes and the system where id_0 is not present at all.

Definition 4 (Anonymity). Given processes ⟨`Register, Issue, Authenticate, Judge`⟩, anonymity is satisfied if the following equivalence holds:

$$(vid.vp.(\texttt{Register}|(p(cre).!(\texttt{Authenticate}|\texttt{Judge})))) \mid$$
$$(let\ id = id_0\ in\ let\ p = int_0\ in\ (\texttt{Register}|(p(cre).!(\texttt{Authenticate}|\texttt{Judge}))))$$
$$\approx$$
$$(!vid.vp.(\texttt{Register}|(p(cre).!(\texttt{Authenticate}|\texttt{Judge}))))$$

Both sides of equivalence are of the same processes except that the left side executes the registration and authentication processes of the user id_0. That is to say, if the equivalence is held, then the adversary cannot tell whether or not the user id_0 has executed the registration and authentication processes.

Unlinkability ensures that a system in which the analyzed processes can be executed by a user multiple times looks the same to an adversary that the system in which the analyzed processes can be executed by the user at most once.

Definition 5 (Unlinkability). Given processes in Definition 4, unlinkability is satisfied if the following equivalence holds:

$$(!vid.vp.(\texttt{Register}|(p(cre).!(\texttt{Authenticate}|\texttt{Judge})))) \approx (!vid.vp.(\texttt{Register}|(p(cre).(\texttt{Authenticate}|\texttt{Judge}))))$$

The difference between two sides locates in the number of times that the authentication has been executed. On condition that this equivalence is satisfied, the adversary cannot distinguish the user executing the authentication multiple times from executing at most once.

3 Case Study: BLACR System

In this section, we model BLACR and automatically verify its security properties using formal analysis tool ProVerif. The review of ProVerif calculus and ZKP compiler are presented in the full version.

3.1 Primitives and Equational Theory

BLACR system employs BBS+ signature scheme, which is proposed by Au *et al.* [9]. In this section, we will introduce the primitives described by applied pi calculus and the associated equational theory.

We consider commitment $\texttt{commit}(x, y)$, where x is a message and y is a commitment factor (or blind factor). We also specify an open function together with the signature scheme for permitting signatures on committed values.

We consider BBS+ signature scheme $\texttt{bbssign}(m, \texttt{sk}(s))$, where m is a message to be signed, and s is a key seed to generate signing key pair $(\texttt{sk}(s), \texttt{pk}(s))$. We specify an open function $\texttt{open}(\texttt{bbssign}(\texttt{commit}(x, y), \texttt{sk}(s)), y)$ for opening the signature of a commitment. Again, we construct a verification function $\texttt{bbsver}(\texttt{open}(\texttt{bbssign}(\texttt{commit}(x, y), \texttt{sk}(s)), y), x, \texttt{pk}(s))$ for this signature. Moreover, a message recovery function $\texttt{getmess}(\texttt{open}(\texttt{bbssign}(\texttt{commit}(\text{x, y}), \texttt{sk}(\text{s})), \text{y}))$ is provided to adversary for getting the signing message x.

We construct a zero-knowledge proof as function $\texttt{ZK}_{i,j}(\widetilde{M}, \widetilde{N}, F)$, where $\widetilde{M}$ is private component representing the knowledge to be proved, $\widetilde{N}$ denote the public component and F denotes a formula over those terms.

In summary, we construct a suitable signature Σ and define an equational theory E to capture the operations of cryptographic primitives. The signature can be defined as follows:

$$\begin{aligned}
&\Sigma = \Sigma_{\text{base}} \cup \Sigma_{ZK}, \text{ where} \\
&\Sigma_{\text{base}} = \left\{ \begin{array}{l} \texttt{true,false,hash,exp,and,or,eq,pk,sk,} \\ \texttt{commit,open,bbssign,bbsver,getmess} \end{array} \right\} \\
&\Sigma_{ZK} = \{\texttt{ZK}_{i,j}, \texttt{Ver}_{i,j}, \texttt{Public}_i, \texttt{Formula}, \alpha_i, \beta_j | i, j \in N\}
\end{aligned}$$

For the signature Σ_{base}, functions `true`, `false` are constant symbols; `hash`, `pk`, `sk`, `getmess` are unary functions; `exp`, `land`, `or`, `eq`, `commit`, `open`, `bbssign` are binary functions; `bbsver` is ternary functions. The equation theory E_{base} associated with signature Σ_{base} is defined as follows:

$$\begin{aligned}
E_{\text{base}} = \; & \texttt{and}(\texttt{true}, \texttt{true}) = \texttt{true} \\
& \texttt{or}(\texttt{true}, x) = \texttt{true} \\
& \texttt{or}(x, \texttt{true}) = \texttt{true} \\
& \texttt{eq}(x, x) = \texttt{true} \\
& \texttt{bbsver}(\texttt{open}(\texttt{bbssign}(\texttt{commit}(x, y), \texttt{sk}(s)), y), x, \texttt{pk}(s)) = \texttt{true} \\
& \texttt{getmess}(\texttt{open}(\texttt{bbssign}(\texttt{commit}(x, y), \texttt{sk}(s)), y)) = x
\end{aligned}$$

Functions `and`, `or`, `eq` are used for conjunction, disjunction and equality test respectively; `hash` is used for hashing messages; `exp` is used for the exponent operation. The rest functions are used for constructing and verifying BBS+ signature scheme.

3.2 Review of BLACR

In this section, we give a high-level description of the BLACR system. The initialization parameters include: the signing key pair $(\texttt{pk}(s_{iss}), \texttt{sk}(s_{iss}))$ of an issuer; the unique identity string *sid* of an SP; the number of categories m and the blacklist of each category with the thresholds TS_i. The registration process proceeds as follows.

1. The issuer sends a random challenge m_{reg} to a user.
2. The user generates a random number y and computes $C_x = \texttt{commit}(x, y)$. Then the user generates a signature proof of knowledge $\Pi_1 = SPK\{(x, y) : C_x = \texttt{commit}(x, y)\}(m_{reg})$. He sends a pair (C_x, Π_1) to the issuer.
3. The issuer computes a blind credential $bcre = \texttt{bbssign}(C_x, \texttt{sk}(s))$ if the verification of Π_1 is successful and then sends *bcre* to the user.
4. The user opens the blind credential $cre = \texttt{open}(bcre, y)$. He outputs *cre* as his credential when the verification $\texttt{bbsver}(cre, x, \texttt{pk}(s))$ is true.

After the user obtains a credential *cre*, he can authentication to the SP multiple times using *cre*. The authentication process is presented below.

1. The SP sends to the user the lists for each category as well as their corresponding threshold values $\widetilde{TS} = (TS_1, ..., TS_m)$ and a random challenge m_{auth} as well as the policy *Pol*.

2. The user judges his reputation score s_i of each category by checking if the entries on the corresponding list belong to him. Then he tests if $s_i < TS_i$ so that Pol evaluates to 1.
3. If the test is successful, the user returns to the SP a pair (τ, Π_2, Π_3), where $\tau = (b, t = H(b||sid)^x)$ is the ticket associated with the current authentication session, and (Π_2, Π_3) is a pair of signature proof of knowledges. Π_2 is used to prove that τ is correctly formed with the credential cre: $SPK\{(x, r, cre) : C_x = \texttt{commit}(x, r), \texttt{bbsver}(cre, x, \texttt{pk}(s)) = \texttt{true}, t = \hat{b}^x\}(m_{auth})$, where $\hat{b} = H(b||sid)$; Π_3 is used to prove Pol evaluates to 1: $SPK\{(x, r, s_i) : C_x = \texttt{commit}(x, r), C_{s_{ij}} = \texttt{commit}(0)|_{j \notin user}, C_{s_{ij}} = \texttt{commit}(s_{ij})|_{j \in user}, C_{s_i} = C_{s_{i1}} \cdot \cdot \cdot C_{s_{iL}}, s_i < TS_i\}(m_{auth})$, where $j \in \{1, ..., L\}$ and L is the length of the corresponding list.
4. The SP verifies the proofs (Π_2, Π_3).

If verification of (Π_2, Π_3) is successful, SP can ensure that the user is a valid one to access the service.

BLACR also realizes a novel approach called express-lane authentication, which can expedite the authentication. To adapt this mechanism, SP should issue a token that is a signature on the aggregated reputation score prior to a time point upon a successful authentication. Then the user can use this token to convince his reputation score in that time instead of proving whether or not an entry belongs to him for every entry in the blacklist. However, using a token disables the SP's capability of unblacklisting since removing entries from blacklist would disable the validity of the token.

3.3 Processes for BLACR

We model the registration phase by a pair of processes $\langle \texttt{Register}, \texttt{Issue} \rangle$ presented in Fig. 1. We assume that every user has a unique id, which can be a limited resource such as IP, mobile phone number to prevent sybil attack. To model the secret value x bound to limited resource, we present a private function `bind` to construct the secret value $x = \texttt{bind}(id)$. We also assume that there is only one category for blacklist, thus there exists only one threshold value TS.

The user first generates a zero-knowledge proof $\Pi_1 = \texttt{ZK}(x, y; id, C_x, m_{reg}; F_{reg})$ to ensure the ownership of x with the formula $F_{reg} = \texttt{and}(\alpha_1 = \texttt{bind}(\beta_1), \beta_2 = \texttt{commit}(\alpha_1, \alpha_2))$. The issuer verifies the validity of Π_1. If the verification is successful, the issuer will check if this user is a sybil. We introduce a predicate $sybil$ in the issuer process. The predicate $sybil$ is true if and only if the user id has been marked sybil. For example, we could set $sybil = \texttt{or}(id = sybilid_1, id = sybilid_2)$ if $sybilid_1, sybilid_2$ have been marked sybil. For a valid id, the issuer signs the commitment C_x and sends the blind signature $bcre$ to the user. The user will open the blind signature $bcre$ and get a credential cre.

We model the authentication phase by a pair of processes $\langle \texttt{Authenticate}, \texttt{Verify} \rangle$ presented in Fig. 2. To generate zero-knowledge proofs, the user must know his reputation score. However, the calculus of ProVerif cannot afford to handle either the algebraic operations or the state transition, so we need a trick

$$
\begin{array}{l}
\texttt{Register} = \\
\quad c(m_{reg}). \\
\quad let\ x = \texttt{bind}(id)\ in \\
\quad vy.let\ C_x = \texttt{commit}(x, y)\ in \\
\quad let\ Pi_1 = \texttt{ZK}(x, y; id, C_x, m_{reg}; F_{reg})\ in \\
\quad \bar{c}\,\langle (C_x, Pi_1)\rangle\,.c(bcre). \\
\quad let\ cre = \texttt{open}(bcre, y)\ in \\
\quad if\ \texttt{bbsver}(cre, x, \texttt{pk}(s_{iss}))\ =\ \texttt{true}\ then \\
\qquad \texttt{event}(\textit{registered}).!\ \bar{p}\,\langle cre\rangle \\
\quad else \\
\qquad \texttt{event}(\textit{unregistered})
\end{array}
\qquad
\begin{array}{l}
\texttt{Issue} = \\
\quad vm_{reg}.\bar{c}\,\langle m_{reg}\rangle\,.c((C_x, Pi_1)). \\
\quad if\ \texttt{public}_2(Pi_1) = C_x\ then \\
\quad if\ \texttt{public}_3(Pi_1) = m_{reg}\ then \\
\quad if\ \texttt{Ver}_{2,3}(F_{reg}, Pi_1) = \texttt{true}\ then \\
\quad let\ id\ = \texttt{public}_1(Pi_1)\ \ in \\
\quad if\ sybil = \texttt{true}\ then\ 0\ else \\
\quad let\ bcre = \texttt{bbssign}(C_x, \texttt{sk}(s_{iss}))\ in \\
\quad \bar{c}\,\langle bcre\rangle
\end{array}
$$

Fig. 1. Process calculus for registration

to model the judgment process. We assume a trusted judgment process outputs the judged score for users. We set the judgment process as follows.

$$
\begin{array}{l}
\texttt{Judge} = asg(s). \\
\quad if\ s = TS\ then\ \texttt{event}(\textit{notSatisfy}).\overline{jud}\,\langle TS\rangle \\
\quad else\ \texttt{event}(\textit{satisfyPolicy}).\overline{jud}\,\langle ltTS\rangle
\end{array}
$$

In a satisfied score judgment, event *satisfyPolicy* is executed and a term $ltTS$ (means less than the threshold value TS) is sent on the private channel jud. Otherwise, event *notSatisfy* is executed and the threshold value TS is sent on the channel jud.

$$
\begin{array}{l}
\texttt{Authenticate} = \\
\quad c(m_{auth}).vr.vb.vr_s \\
\quad let\ x = \texttt{bind}(id)\ in \\
\quad let\ C_x = \texttt{commit}(x, r)\ in \\
\quad let\ h = \texttt{hash}((b, sid))\ in \\
\quad let\ t = \texttt{exp}(h, x)\ in \\
\quad let\ Pi_2 = \texttt{ZK}(x, r, cre; C_x, \texttt{pk}(s_{iss}), b, t, h, \\
\qquad m_{auth}; F_{sig})\ in \\
\quad jud(s).let\ C_s = \texttt{commit}(s, r_s)\ in \\
\quad let\ Pi_3 = \texttt{ZK}(x, r, s, r_s; C_x, C_s, ltTS, m_{auth}; \\
\qquad F_{Pol})\ in \\
\quad \texttt{event}(\textit{startAuth}).\bar{c}\,\langle (Pi_2, Pi_3)\rangle
\end{array}
\qquad
\begin{array}{l}
\texttt{Verify} = \\
\quad vm_{auth}.\bar{c}\,\langle (v_{auth})\rangle\,.c((Pi_2, Pi_3)). \\
\quad if\ \texttt{public}_2(Pi_2) = \texttt{pk}(s_{iss})\ then \\
\quad if\ \texttt{public}_5(Pi_2) = \texttt{hash}((\texttt{public}_3(Pi_2), \\
\qquad sid))\ then \\
\quad if\ \texttt{public}_6(Pi_2) = m_{auth}\ then \\
\quad if\ \texttt{Ver}_{3,6}(F_{sig}, Pi_2) = \texttt{true}\ then \\
\quad let\ C_x = \texttt{public}_1(Pi_2)\ in \\
\quad let\ b = \texttt{public}_3(Pi_2)\ in \\
\quad let\ t = \texttt{public}_4(Pi_2)\ in \\
\quad if\ \texttt{public}_1(Pi_3) = C_x\ then \\
\quad if\ \texttt{public}_3(Pi_3) = ltTS\ then \\
\quad if\ \texttt{public}_4(Pi_3) = m_{auth}\ then \\
\quad if\ \texttt{Ver}_{4,4}(F_{Pol}, Pi_3) = \texttt{true}\ then \\
\qquad \texttt{event}(\textit{acceptAuth}).!\ \overline{lt}\,\langle (b, t)\rangle \\
\quad else \\
\qquad \texttt{event}(\textit{revoke})
\end{array}
$$

Fig. 2. Process calculus of authentication

Then the user generates two zero-knowledge proofs: Π_2 with formula $F_{sig} = \texttt{and}(\texttt{and}(\beta_1 = \texttt{commit}(\alpha_1, \alpha_2), \texttt{bbsver}(\ \alpha_3, \alpha_1, \beta_2) = \texttt{true}), \beta_4 = \texttt{exp}(\beta_5, \alpha_1))$

and Π_3 with formula $F_{Pol} = \texttt{and}(\texttt{and}(\beta_1 = \texttt{commit}(\alpha_1, \alpha_2), \beta_2 = \texttt{commit}(\alpha_3, \alpha_4)), \beta_3 = \alpha_3)$. The process executes the event *startAuth* before it outputs $\langle \Pi_2, \Pi_3 \rangle$.

The process of SP verifies $\langle \Pi_2, \Pi_3 \rangle$ and executes the event *acceptAuth* when all verifications are passed, otherwise, it executes the event *revoke*. For a successful authentication, this process also "stores" the ticket (b, t) of current authentication by a private channel lt for further assigning reputation score.

3.4 Experiment Results

In this section, we examine the security properties defined in Sect. 2 using ProVerif. The processes above will be expressed as specifications of ZKP compiler and then be encoded into the inputs of ProVerif. The detailed specifications can be found in [28].

Security Properties as Correspondences. Security goals *Authenticity*, *Non-frameability* and *Mis-authentication resistance* are expressed by correspondences. To verify these properties, we implement a main process `C-Process` as follows:

$$\begin{aligned}&\texttt{C-Process}\\ &\quad vs_{iss}.vs_{ver}.vjud.vlt.let\ c = pub\ in\ (\overline{pub}\,\langle \texttt{pk}(s_{iss})\rangle \mid \overline{pub}\,\langle \texttt{pk}(s_{ver})\rangle \mid\\ &\quad (!\,\texttt{issue}) \mid (!\,(\texttt{Verify} \mid \texttt{AssignScore})) \mid \texttt{ControlUsers})\end{aligned}$$

Note that we also initialize a key pair for the SP since we set $sid = \texttt{pk}(s_{ver})$ to identify the SP for computing the ticket.

Result 1. *Given the main process* `C-Process`*,* ProVerif *succeeds in proving the correspondence statements defined in Sect. 2.2. Hence, security properties Authenticity,Non-frameability and Mis-authentication resistance are held.*

Security Properties as Equivalence. Privacy of BLACR is expressed by biprocess. We identify two kinds of privacy: *anonymity* and *unlinkability*. We implement a main process `A-Process` to capture anonymity.

$$\begin{aligned}&\texttt{A-Process}\\ &\quad vs_{iss}.vs_{ver}.vjud.vlt.vint_0.vint_1.let\ c = pub\ in\ (\\ &\quad \overline{pub}\,\langle \texttt{pk}(s_{iss})\rangle \mid \overline{pub}\,\langle \texttt{pk}(s_{ver})\rangle \mid (!\texttt{Issue}) \mid \texttt{Users}\mid\\ &\quad (let\ (id,p) = (id_0, int_0)\ in\ \texttt{Register}) \mid (vid_1.let\ (id,p) = (id_1, int_1)\ in\ (\\ &\qquad \texttt{Register}\mid int_0(cre_0).int_1(cre_1).\\ &\qquad let\ (id, cre) = (\texttt{diff}[id_0, id_1], \texttt{diff}[cre_0, cre_1])\ in\ (\texttt{Authenticate}\mid\texttt{Judge})))\\ &)\\ &where\\ &\quad \texttt{Users} = !vid.!vp.(\texttt{Register}\mid(p(cre).!(\texttt{Authenticate}\mid\texttt{Judge})))\end{aligned}$$

To adapt the definition of anonymity, we use the process `Users` instead of `ControlUsers` to capture arbitrary users except id_0 executing the processes.

Encoding anonymity in this way, we have the left side of `diff` representing an execution of publicly known id id_0, while the right side of `diff` represents an execution of unknown id id_1 (a restrict id). In fact, the right side of `diff` is a case of `Users`. Hence, it directly corresponds to the definition in Sect. 2.2 and we succeed in reducing the problem of proving anonymity to the diff-equivalence that can be verified by ProVerif.

Result 2. *Given the main process* `A-Process`, ProVerif *succeeds in proving the diff-equivalence, therefore, anonymity is satisfied.*

We also implement a main process `U-Process` to capture unlinkability.

`U-Process`
$vs_{iss}.vs_{ver}.vjud.vL.let\ c = pub\ in\ (\overline{pub}\,\langle \mathtt{pk}(s_{iss})\rangle\ \ |\overline{pub}\,\langle \mathtt{pk}(s_{ver})\rangle\ |$
$(!\ \mathtt{issue})\ |\mathtt{Unlinkability})$

where

$\mathtt{Unlinkability} =$
$!vid_1.vint_1.((let\ (id,p) = (id_1, int_1)\ in\ \mathtt{Register})|(!vid_2.vint_2.($
$(let\ (id,p) = (id_2, int_2)\ in\ \mathtt{Register})|(int_1(cre_1).int_2(cre_2).$
$let\ (id,cre) = (\mathtt{diff}[id_1,id_2],\mathtt{diff}[cre_1,cre_2])\ in\ (\mathtt{Authenticate}|\mathtt{Judge}))\ ))$
$)$

Thinking inside this process, we have that the left side of the `diff` representing a user executes the system many time, while the right side represents the users execute the system at most once (The user id_2 is always different for each execution of processes of the user id_1).

Result 3. *Given the main process* `U-Process`, ProVerif *succeeds in proving the diff-equivalence, therefore, unlinkability is satisfied.*

3.5 Express-Lane Case in BLACR

To reward active users, BLACR offers express-lane authentication to speed up the authentication process. In the express-lane authentication, an SP additionally signs a credential (a token) on the score of previous time after the verification succeeds. This token will be used in the next authentication.

However, an additional state transition that ProVerif cannot deal with is introduced since a token generated by current authentication must be transferred to the next time for use. Hence we have to bring in another trick to adapt ProVerif. We divide the analysis into two scenarios: the first is the one that a user is honest people, thus always getting a valid token and proceeding as expectation; the second is the one that a user will be revoked and test if BLACR proceeds as expectation.

Since the processes are similar to the normal one, they will be omit for the space limitation. Specifically, we have presented the details in the full version.

Result 4. ProVerif *discovers an attack trace in the second express-lane authentication. As a consequence, we say that the token mechanism of BLACR does not work properly.*

The attack trace shows that a replay attack can be carried out by a malicious user as follows: in the second express-lane authentication, the user finds his aggregated reputation score does not satisfy the authentication policy. But he still proceeds in this way: he uses a preceding token that is enough to make the aggregate score satisfying the policy. This attack can happen since these tokens do not consist of any labels to distinguish each other.

In general, this attack can be applied to two scenarios violating the security policy: the first one is that a user can utilize an old token to escape from being revoked; the other is that a user in possession of a token can conduct an express-lane authentication at any time, regardless of whether he is an active user.

This attack can be fixed by refining the definition of token tk. The token must consist of the timestamp information t. We revise the processes with the timestamp and the verification is successful by ProVerif.

In fact, our solution in symbolic representation mode indicates that the fix presented in ExBLACR still does not work properly since the timestamp in the proving process of ExBLACR does not be revealed. In such way, a malicious user can still conduct the replay attack mentioned above, because the SP can just ensure the token tk is correct but can not know the timestamp t corresponding to this token.

4 Conclusion

This paper presents the definitions of some common security properties for BLAC-like systems in the symbolic model using applied pi calculus. We express these definitions as correspondence and equivalence properties. As a case study, we verify these properties in BLACR system. The analysis finds a known attack aiming at the token mechanism in the express-lane authentication. This revision with a successful verification in ProVerif also indicates that the fix provided by ExBLACR is incorrect.

Actually, our model is of approximate due to the nature of ProVerif. We think some other modelling method can also be under consideration to record state, such as multiset rewriting rules (Tamarin tool [24]) or stateful variant of applied pi calculus [5,17]. Another extension may be lying in research of composing protocols as mentioned in the introduction. These may be the future work.

Acknowledgements. The research presented in this paper is supported by the National Grand Fundamental Research 973 Program of China under Grant No.2013CB338003 and the National Natural Science Foundation of China under Grant Nos. 91118006, 61202414.

References

1. Abadi, M., Blanchet, B.: Analyzing security protocols with secrecy types and logic programs. J. ACM (JACM) **52**(1), 102–146 (2005)
2. Arapinis, M., Cheval, V., Delaune, S.: Verifying privacy-type properties in a modular way. In: CSF 2012, pp. 95–109. IEEE (2012)
3. Arapinis, M., Cheval, V., Delaune, S.: Composing security protocols: from confidentiality to privacy. In: Focardi, R., Myers, A. (eds.) POST 2015. LNCS, vol. 9036, pp. 324–343. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46666-7_17
4. Arapinis, M., Chothia, T., Ritter, E., Ryan, M.: Analysing unlinkability and anonymity using the applied pi calculus. In: CSF 2010, pp. 107–121. IEEE (2010)
5. Arapinis, M., Phillips, J., Ritter, E., Ryan, M.D.: Statverif: verification of stateful processes. J. Comput. Secur. **22**(5), 743–821 (2014)
6. Au, M.H., Tsang, P.P., Kapadia, A.: PEREA: practical TTP-free revocation of repeatedly misbehaving anonymous users. ACM Trans. Inf. Syst. Secur. (TISSEC) **14**(4), 29 (2011)
7. Au, M.H., Kapadia, A.: PERM: practical reputation-based blacklisting without TTPs. In: CCS 2012, pp. 929–940. ACM (2012)
8. Au, M.H., Kapadia, A., Susilo, W.: BLACR: TTP-free blacklistable anonymous credentials with reputation. In: NDSS Symposium 2012: 19th Network & Distributed System Security Symposium, pp. 1–17. Internet Society (2012)
9. Au, M.H., Susilo, W., Mu, Y.: Constant-size dynamic k-TAA. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 111–125. Springer, Heidelberg (2006). https://doi.org/10.1007/11832072_8
10. Backes, M., Maffei, M., Unruh, D.: Zero-knowledge in the applied pi-calculus and automated verification of the direct anonymous attestation protocol. In: SP 2008, pp. 202–215. IEEE (2008)
11. Blanchet, B.: Automatic proof of strong secrecy for security protocols. In: SP 2004, pp. 86–100. IEEE (2004)
12. Blanchet, B., Abadi, M., Fournet, C.: Automated verification of selected equivalences for security protocols. In: 20th IEEE Symposium on Logic in Computer Science. Proceedings, pp. 331–340. IEEE (2005)
13. Blanchet, B., et al.: ProVerif: cryptographic protocol verifier in the formal model. http://prosecco.gforge.inria.fr/personal/bblanche/proverif/
14. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: Proceedings of the 11th ACM Conference on Computer and Communications Security, pp. 132–145. ACM (2004)
15. Brickell, E., Chen, L., Li, J.: A new direct anonymous attestation scheme from bilinear maps. In: Lipp, P., Sadeghi, A.-R., Koch, K.-M. (eds.) Trust 2008. LNCS, vol. 4968, pp. 166–178. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68979-9_13
16. Brickell, E., Li, J.: Enhanced privacy ID: a direct anonymous attestation scheme with enhanced revocation capabilities. In: Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society, pp. 21–30. ACM (2007)
17. Bruni, A., Modersheim, S., Nielson, F., Nielson, H.R.: Set-pi: set membership p-calculus. In: CSF 2015, pp. 185–198. IEEE (2015)
18. Camenisch, J., Drijvers, M., Lehmann, A.: Universally composable direct anonymous attestation. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016. LNCS, vol. 9615, pp. 234–264. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49387-8_10

19. Camenisch, J., et al.: Specification of the identity mixer cryptographic library, version 2.3. 1, 7 December 2010
20. Cheval, V., Blanchet, B.: Proving more observational equivalences with ProVerif. In: Basin, D., Mitchell, J.C. (eds.) POST 2013. LNCS, vol. 7796, pp. 226–246. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36830-1_12
21. Johnson, P.C., Kapadia, A., Tsang, P.P., Smith, S.W.: Nymble: anonymous IP-address blocking. In: Borisov, N., Golle, P. (eds.) PET 2007. LNCS, vol. 4776, pp. 113–133. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75551-7_8
22. Paquin, C., Zaverucha, G.: U-prove cryptographic specification v1. 1. Technical report, revision 3. Microsoft Corporation (2013)
23. Ryan, M.D., Smyth, B.: Applied pi calculus. In: Cortier, V., Kremer, S. (eds.) Formal Models and Techniques for Analyzing Security Protocols, Chap. 6. IOS Press (2011)
24. Schmidt, B., Meier, S., Cremers, C., Basin, D.: Automated analysis of Diffie-Hellman protocols and advanced security properties. In: CSF 2012, pp. 78–94. IEEE (2012)
25. Smyth, B., Ryan, M., Chen, L.: Formal analysis of anonymity in ECC-based direct anonymous attestation schemes. In: Barthe, G., Datta, A., Etalle, S. (eds.) FAST 2011. LNCS, vol. 7140, pp. 245–262. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29420-4_16
26. Smyth, B., Ryan, M.D., Chen, L.: Formal analysis of privacy in direct anonymous attestation schemes. Sci. Comput. Program. **111**, 300–317 (2015)
27. Tsang, P.P., Au, M.H., Kapadia, A., Smith, S.W.: BLAC: revoking repeatedly misbehaving anonymous users without relying on TTPs. ACM Trans. Inf. Syst. Secur. (TISSEC) **13**(4), 39 (2010)
28. Wang, W.: Proverif inputs for analyzing BLACR system. https://github.com/WangWeijin/Formal-analysis-of-BLACR-system
29. Wang, W., Feng, D., Qin, Y., Shao, J., Xi, L., Chu, X.: ExBLACR: extending BLACR system. In: Susilo, W., Mu, Y. (eds.) ACISP 2014. LNCS, vol. 8544, pp. 397–412. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08344-5_26
30. Xi, L., Feng, D.: FARB: fast anonymous reputation-based blacklisting without TTPs. In: Proceedings of the 13th Workshop on Privacy in the Electronic Society, pp. 139–148. ACM (2014)
31. Xi, L., Feng, D.: Formal analysis of DAA-related APIs in TPM 2.0. In: Au, M.H., Carminati, B., Kuo, C.-C.J. (eds.) NSS 2014. LNCS, vol. 8792, pp. 421–434. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11698-3_32
32. Yu, K.Y., Yuen, T.H., Chow, S.S.M., Yiu, S.M., Hui, L.C.K.: PE(AR)^2: privacy-enhanced anonymous authentication with reputation and revocation. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012. LNCS, vol. 7459, pp. 679–696. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33167-1_39