Refresh Instead of Revoke Enhances Safety and Availability: A Formal Analysis

Mehrnoosh Shakarami(✉) and Ravi Sandhu

Institute for Cyber Security (ICS), Center for Security and Privacy Enhanced Cloud Computing (C-SPECC), Department of Computer Science, University of Texas at San Antonio, San Antonio, USA
mehrnoosh.shakarami@my.utsa.edu, ravi.sandhu@utsa.edu

Abstract. Due to inherent delays and performance costs, the decision point in a distributed multi-authority Attribute-Based Access Control (ABAC) system is exposed to the risk of relying on outdated attribute values and policy; which is the safety and consistency problem. This paper formally characterizes three increasingly strong levels of consistency to restrict this exposure. Notably, we recognize the concept of refreshing attribute values rather than simply checking the revocation status, as in traditional approaches. Refresh replaces an older value with a newer one, while revoke simply invalidates the old value. Our lowest consistency level starts from the highest level in prior revocation-based work by Lee and Winslett (LW). Our two higher levels utilize the concept of request time which is absent in LW. For each of our levels we formally show that using refresh instead of revocation provides added safety and availability.

Keywords: ABAC · Refresh · Consistency · Safety · Availability

1 Introduction

In Attribute-Based Access Control (ABAC), access decisions are made based on attribute values of subjects, objects and environment with respect to a given policy. Attribute values for subjects and objects are typically provisioned by an Attribute Authority (AA) and presented in credentials as name, value pairs. A credential must be trustworthy, perhaps by a cryptographic signature or trusted delivery. Attribute values are susceptible to change. Ideally the decision point should know real-time values, which is practically impossible due to inherent delays of distributed systems and performance costs. This can lead to granting access when it should be denied (safety violation) or denying access when it should be granted (availability violation). The longer the gap between updates of credentials, the higher the risk of relying on stale attribute values.

Published by Springer Nature Switzerland AG 2019
S. N. Foley (Ed.): DBSec 2019, LNCS 11559, pp. 301–313, 2019.
https://doi.org/10.1007/978-3-030-22479-0_16

In this paper we formally characterize three increasingly strong levels of consistency to restrict the exposure of the decision point to stale attribute values. For simplicity, we develop our formalism based on changing subject attribute values. Extension to changing object and environment attribute values is straightforward. Extension to policy changes is more subtle. Policy changes may require additional credentials come into play. While acquiring these additional credentials the policy may change again. In principle, this could lead to an infinite regress. In practice such an infinite regress is unlikely. Policies composed of multiple sub-policies specified by different authorities also raise issues of policy conflicts [5,17]. A formal treatment of policy changes is beyond our scope.

The closest prior work is by Lee and Winslett (LW) [15,16]. Our paper is inspired by LW but presents a completely new perspective by considering refresh instead of revocation. We build our levels on top of the highest consistency level of LW, recasting it in the refresh framework. Taking request time into account, we propose two higher consistency levels not available in LW.

Our main contribution is to develop a formal framework for safety, availability and consistency problems of ABAC systems, via introducing the refresh scenario instead of the traditional revocation check. As we will show, this enhanced possibility of getting a new value rather than an invalid response enhances safety and availability. We also define the concept of being `satisfactory` for an attribute value with respect to a policy, which is first introduced in our work to the best of our knowledge. Relying on the history of `satisfactory` attribute values, we introduce additional flexibility to grant access to authorized users.

The paper is organized as follows. A review of related work and a comparison to LW is given in Sect. 2. Section 3 documents our system model and assumptions. The formalism of our consistency levels along with guaranteed properties by each specification is given in Sect. 4. Limitations and practical implications are discussed in Sect. 5. Section 6 concludes the paper.

2 Related Work

There is a rich body of research work on consistency in distributed systems [1,2,8,20,24]. Many access control models are not completely compatible with distributed systems in that they are not deployed for such systems in the first place [10]. ABAC is well adjusted to distributed environments due to its flexibility and granularity. In this paper, we consider an ABAC model to be in place and define consistency as communicating credentials' updates as quickly as possible to the decision point. To the best of our knowledge there is very limited directly related research in this arena. Especially there is no work done toward utilizing the refresh operation to obtain recent information.

The closest to our research is LW in trust negotiation environments [14–16]. Another closely related research is on stale-safety which tries to safely uses stale attributes [12,13]. Although the problem is similar, it mainly differs from our work since it has been applied in a non-ABAC, single authority environment. Policy staleness of cloud transactions proposed in [11].

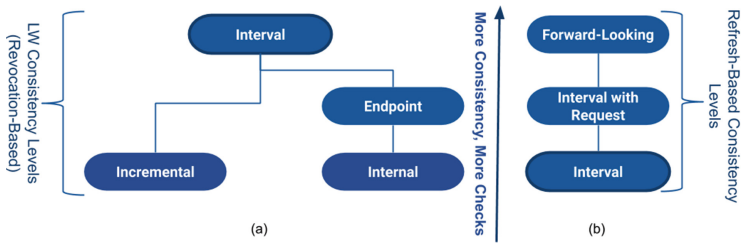

Fig. 1. (a) LW revocation-based levels [15,16] (b) Our refresh-based levels

Ciphertext-Policy Attribute-Based Encryption [3,4] is broadly applicable in decentralized multi-authority environments, but presents challenge to handle attribute revocation [22,26–28]. Moreover, it imposes a heavy performance burden which makes it impractical [6]. There are other researches concerning the policy consistency in distributed environments [6,11,18,29] focusing on cloud environments. In this paper, policy assumed to be known with high assurance. There are research works utilizing revocation in authenticated dynamic dictionaries [7,21,25], which enable dissemination of information from a secure central repository to multiple recepients.

Comparison to LW Model. LW presented the first organized work on consistency in trust negotiation systems. They proposed four consistency levels based on timeliness of credentials revocation checks. In common with our model they considered every credential to have its lifetime specified by start time and end time. While all levels in LW model utilize the notion of receive time of credentials, we are agnostic to it. We consider decision time as central and utilize it explicitly in all levels, whereas in LW it is explicit only in top two levels. Revocation check in LW is replaced with refresh in our model, as will be discussed in next section. An alternate formulation of LW without use of receive time is given in [23], which includes an additional level based on request time.

Figure 1-(a) shows the levels in LW which are partially ordered. We do not recommend using incremental and internal levels since in both cases decision point may use a credential which is known to be expired or revoked. Our proposed levels are shown in Fig. 1-(b) with a total order among levels. We set LW's highest level as the base level in our definitions. By taking request time into account, we propose two additional stronger levels of consistency. We provide further availability in our model by letting the decision point consider valid authorization should the current and cached values of relevant credentials be `satisfactory`, as defined in following sections.

3 System Model and Assumptions

We assume an ABAC authorization system in a distributed multi-authority environment. For a particular access request, there is a single decision point which

determines whether or not the access is allowed by the access control policy based on attribute values. For convenience we use the terms attribute and attribute value interchangeably. The main focus of this paper is to limit the exposure of the decision point to outdated attributes by enforcing timeliness of checking subjects' attributes freshness.

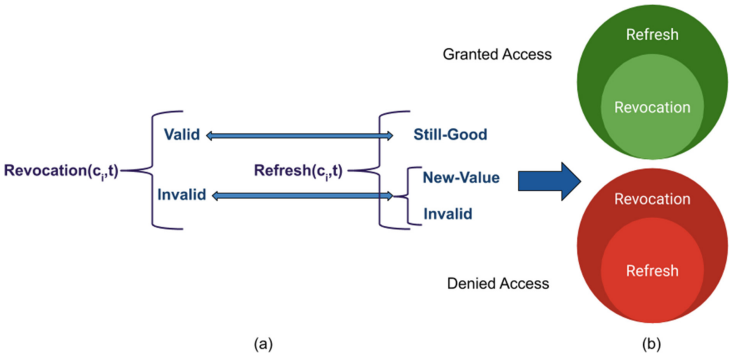

Fig. 2. (a) Revocation vs. Refresh (b) Comparing Grant vs. Deny

Table 1. Summary table of symbols

Symbol	Meaning
t_{req}	request time
t_d	decision time
c_i	i^{th}credential
t^i_{revoc}	actual revocation time of c_i (the AA always knows this time)
$t^i_{ref,k}$	time of k-th refresh of c_i
$t^i_{start,k}$	attribute start time of c_i after k-th refresh
$t^i_{end,k}$	attribute expiration time of c_i after k-th refresh
$kmax(t)$	latest refresh of c_i before time t (c_i is determined by context)
$val^i_{kmax(t)}$	the value of c_i after $kmax(t)$-th refresh
$t^i_{ref,kmax(t)}$	time of $kmax(t)$-th refresh of c_i
$t^i_{start,kmax(t)}$	attribute start time of c_i after $kmax(t)$-th refresh
$t^i_{end,kmax(t)}$	attribute expiration time of c_i after $kmax(t)$-th refresh

3.1 Refresh Vs. Revocation

Subject attributes might change during credential lifetime. A change could be a new value, a new lifetime or a premature revocation. In all cases the decision point needs to be updated about the latest changes of the attribute through

either revocation or refresh. In revocation, AA would represent the current status of the credential as either `Valid` (no change) or `Invalid` (otherwise). However, with refresh AA can indicate the credential's status as `Still-Good`, `New-Value` or `Invalid`. `Still-Good` and `Invalid` correspond to `Valid` or `Invalid` in revocation scenario. `New-Value` reflects any change in credential's start time, end time or new value. So, `Invalid` status in revocation splits in two possibilities of `Invalid` and `New-Value` in refresh (see Fig. 2-a). Thereby, refresh can allow more accesses than revoke and deny fewer accesses (see Fig. 2-b).

Refresh function is defined as follows. T is the set of possible time stamps and C represents the set of all credentials in the system. Table 1 defines the symbols used in this definition and throughout the paper.

$$Refresh : C \times T \rightarrow \{Invalid, Still\text{-}Good, New\text{-}Value\} \tag{1}$$

$$Refresh(c_i, t) = \begin{cases} Invalid & \iff (t \geq t^i_{end,kmax(t)}) \vee (t \geq t^i_{revoc}) \\ New\text{-}Value & \iff (t^i_{start,kmax(t)} \neq t^i_{start,kmax(t)-1}) \\ & \vee (t^i_{end,kmax(t)} \neq t^i_{end,kmax(t)-1}) \vee (val^i_{kmax(t)} \neq val^i_{kmax(t)-1}) \\ Still\text{-}Good & \iff (t^i_{start,kmax(t)} = t^i_{start,kmax(t)-1}) \\ & \wedge (t^i_{end,kmax(t)} = t^i_{end,kmax(t)-1}) \wedge (val^i_{kmax(t)} = val^i_{kmax(t)-1}) \end{cases}$$

Following example highlights the benefits provided by considering refresh rather than revocation. Although granting illegitimate access is considered as a greater risk in many systems, availability is also important in which a legitimate user should not be denied access.

Example 1. Authorization policy in a coding company grants read access to a project's code to managers and test engineers and read/write access to developers. Alice was a test engineer. But her role has changed to a developer in the same project. Subsequently she submits a write request to the decision point. In revocation, checking her cached role credential results in `Invalid` response since she is no longer a test engineer. So her request would be denied. In refresh, however, `New-Value` response along with a new credential asserting her new role would be returned and access would be granted, as it should be based on policy.

Claim. If a subject can proceed to utilize a requested access in a revocation scenario, it can proceed in a refresh scenario as well. But there are scenarios in refresh-based systems which let the subject proceed, whereas it would be denied in revocation-based systems.

Proof. If nothing changed about a required credential, revocation and refresh would return `Valid`/`Still-Good` respectively. So, the first part of the claim follows. For the second part, it is possible that a required credential has changed with respect to start/end time or the value. So AA response in revocation scenario will be `Invalid` which prohibits subject's access. However with refresh the response would be `New-Value`, so access would be granted (see Fig. 2).

3.2 System Assumptions

Without loss of generality, we suppose that the policy is stated in Disjunctive Normal Form (DNF), which is the disjunction of different conjuncts. The decision point tries to find the first conjunct which satisfies the desired level of consistency. This conjunct is called the `View` of the decision point at any specific time t with respect to the policy P which we denote as $V_{DP}^{P,t}$. We assume the decision point can instantaneously check the policy and identify the view.

Definition 1. *At any time t, we call the set of subject's attributes included in* $V_{DP}^{P,t}$ *as the relevant credentials.*

We make following assumptions in this paper.

1. Attributes do not change as the result of attribute credentials usage, that is we assume attributes to be immutable in sense of [19].
2. We will not utilize any expired credential. If any required credential is beyond its end time, decision point polls AA to get a new credential for the attribute.
3. We do not refresh any credential after it has been found to be `Invalid`.
4. There is one instantaneous decision time (t_d) and one instantaneous request time (t_{req}).
5. V_{DP}^{P,t_d} is the only view of our interest as described above.
6. If refresh returns a `New-Value` result, its start time cannot be prior to its previous start time, i.e., $t_{start,k}^{i} \geq t_{start,k-1}^{i}$.
7. AA will not return a credential along with `New-Value` which has not been started yet, so, $t_{ref,k}^{i} \geq t_{start,k}^{i}$.

4 Consistency Levels Formal Characterization

4.1 Preliminaries

Satisfactory Values. We define an attribute to be `satisfactory` if and only if its value fulfills the policy conditions. For instance if the policy requires the security level to be at least 3, any security level credential with the value greater than or equal to 3 is considered as `satisfactory`. Obviously the same credential may not be `satisfactory` with respect to another policy. We formally define `satisfactory` with respect to a policy P at the specific time t as follows.

Definition 2. *The view at time t has the structure* $V_{DP}^{P,t} = \bigwedge_{1 \leq i \leq n} F(i)$ *in which* $F(i)$ *is an atomic expression specifying required conditions for* c_i*'s value. We define Sat as follows to determine satisfactory requirements for* c_i*'s value.*

$$Sat_{c_i}^{P,t} = True \iff F(val_{kmax(t)}^{i}) = True \tag{2}$$

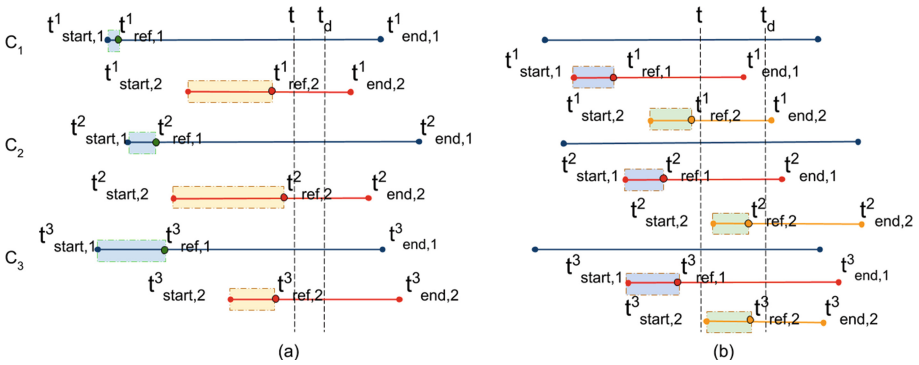

Fig. 3. Interval consistency

Freshness. We rely on the freshness concept in refresh scenario, compared to validity in revocation scenario. We formally define freshness via *Fresh* function as follows. When *Fresh* is used in a boolean expression, we understand $Fresh(c_i, t)$ to be *False* when its value is *Unknown*.

$$Fresh : C \times T \rightarrow \{True, False, Unknown\}$$

$$Fresh(c_i, t) = \begin{cases} True & \iff (t^i_{start,k} \leq t \leq t^i_{ref,k}) \\ & \wedge (Refresh(c_i, t^i_{ref,k}) \neq Invalid) \\ Unknown & \iff (t^i_{ref,k} < t < t^i_{end,k}) \vee (t \geq t^i_{ref,kmax(t)}) \\ False & \iff [(t \geq t^i_{ref,k}) \wedge (Refresh(c_i, t^i_{ref,k} = Invalid))] \\ & \vee [t \geq t^i_{end,kmax(t)}]) \end{cases} \tag{3}$$

Following example is used throughout the paper.

Example 2. In a company, project managers and testing engineers with the security level of at least 5 can access project's documents. The policy in DNF form is $P = [(role \in \{manager, engineer\}) \wedge (security\text{-}level \geq 5)]$. Bob is a project manager since January 1st to January 25th based on a refresh at January 15th. A refresh at January 21st shows his role has changed to testing engineer as of January 20th through March 20th. A refresh at January 15th shows his security level is 6 as of January 10th to March 20th. Another refresh at January 28th reveals security level has been downgraded to 4 since January 26th through March 20th.

We now introduce three levels of consistency taking both old and new values of relevant credentials into account. We provide specifications and consequent properties guaranteed by each level in the rest of this section.

4.2 Interval Consistency

At this level, it is required to find overlap of freshness intervals (simultaneous freshness) of relevant credentials before the decision time. In Example 2, suppose

Bob requests access to project documents on Jan 18th. Based on refresh results at Jan 15th, decision point finds simultaneous freshness of relevant credentials during Jan 10th-Jan 15th with satisfactory values. So the access will be granted. The stipulated overlap could be found for most recent refresh results of relevant credentials (Fig. 3-(a)) or by considering both old and new refresh results (Fig. 3-(b)). In these and subsequent figures if any refresh shown on the first line, it returns `Still-Good` while any other refresh returns `New-Value`. Moreover, in all cases the values of the three credentials are `satisfactory`. In Fig. 3-(a) the overlap is for the most recent refreshed values, whereas in Fig. 3-(b) the overlap is for a mix of the refreshed values, one new and two older.

Specification. Every credential has been refreshed at least once before the decision time and found to be fresh. Most recent values of all relevant credentials are satisfactory with respect to the policy. Any overlap of freshness intervals for the freshest/cached credentials is acceptable so long as the values are satisfactory.

$$
\begin{aligned}
&Interval(V_{DP}^{P,t_d}) \iff (\exists t \leq t_d)(\forall c_i \in V_{DP}^{P,t_d}) \\
&[\max_{\forall c_j \in V_{DP}^{P,t_d}} t^j_{start,kmax(t)} \leq t^i_{ref,kmax(t)} < \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{end,kmax(t)} \\
&\wedge Fresh(c_i, t^i_{ref,kmax(t)}) \wedge Fresh(c_i, t^i_{ref,kmax(t_d)}) \wedge Sat^{P,t}_{c_i} \wedge Sat^{P,t_d}_{c_i} \\
&\wedge \max_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{start,kmax(t_d)} < t_d < \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{end,kmax(t_d)}]
\end{aligned}
\tag{4}
$$

Property 1. There is a time interval during which all relevant credentials were simultaneously fresh with satisfactory values with respect to the policy.

Proof. Based on Eq. (4), there exists a time (t) prior to the decision time at which the latest refresh of every relevant credential happens after all have been started and before any of them ends. This implies all credentials are simultaneously fresh during $[\max_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{start,kmax(t)}, \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{ref,kmax(t)}]$.

Corollary 1. if $t = t_d$, latest values of relevant attributes have freshness overlap.

Comparing with Revocation-Based Scenario. Based on the Claim in Sect. 3.1, revocation and refresh are the same in case of `Valid` and `Still-Good` responses from AA. But if the result is `New-Value`, the corresponding revocation result would be `Invalid` which denies the access. In Example 2, if Bob requests access to the project's documents on Jan 25th and decision point rechecks the credentials, although Bob's role has changed, he would get the access in the refresh scenario whereas he would be denied in revocation scenario.

4.3 Interval Consistency with Request Time

In first level, the decision point relies on what avails of previous refresh results for relevant credentials and access would be denied in case of any unrefreshed

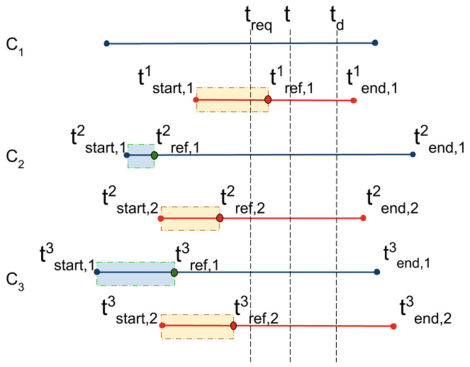

Fig. 4. Interval consistency with request time

credential. By considering the request time we could compensate for missing refreshes. In Example 2, if Bob requests for accessing project's documents on January 14th, the access would be denied at first level since there is no refresh result available for required credentials. At second level, the decision point refreshes the credentials after the request time and then checks the consistency requirements. Figure 4 shows similar example where the top credential is refreshed after request time.

Specification. Decision point refreshes any credential with missing refresh results after the request time. Afterwards, relevant credentials should satisfy the interval consistency (previous level) requirements.

$$\begin{aligned} &IntervalWithReq(V_{DP}^{P,t_d}) \iff (\forall c_i \in V_{DP}^{P,t_d})\ [t^i_{ref,kmax(t_{req})} \neq \perp \\ &\vee (\exists t_r\ \ t_{req} < t_r < t_d)\ \ Refresh(c_i, t_r)] \wedge Interval(V_{DP}^{P,t_d}) \end{aligned} \tag{5}$$

Proposition 1. *We assume the set of relevant credentials would not change during the short gap between request time and decision time, so,* $V_{DP}^{P,t_{req}} = V_{DP}^{P,t_d}$. *In other words the policy will not frequently change in the system.*

Property 1. There is a time interval during which all relevant credentials are simultaneously fresh. Possible lack of refresh would not unnecessarily deny access.

Proof. Use of the same requirement of $Interval(V_{DP}^{P,t_d})$ guarantees the same property of freshness overlap of relevant credentials. Any missing refresh results would be compensated after request time. It is possible that the gap between the request time and decision time does not last enough to compensate all the lacking information, but we consider it as an administrative setting which is out of scope for this paper to quantify.

Property 2. Every interval consistent view with request time satisfies the interval consistency requirements as well.

Proof. The proof is trivial since this level is defined based on interval level.

Property 3. An interval consistent view may deny access allowed by interval consistent with request time.

Proof. Since we do not consider request time in first level, there is no opportunity to compensate possible missing refreshes which could enable access.

Comparing with Revocation-Based Scenario. Considering the formal specification in Eq. (5), which is based on first level, the comparison is trivial. If refresh is substituted with revocation, system's availability would decrease as discussed in Sect. 3.1. The same situation may happen with regard to Example 2 as discussed in Sect. 4.2.

4.4 Forward-Looking Consistency

This level provides simultaneous freshness of all relevant credentials *after the request time*, considering both new and old credentials. Overlapping interval could either include the request time (Fig. 5-(a)) or not (Fig. 5-(b)). In Example 2, if Bob requests access to project's documents on Feb 1st his credentials would be refreshed afterwards revealing changes in role and security level leading to deny. However in previous levels an unauthorized access may be granted.

Specification. Any relevant credential has to be refreshed at least once after the request time. All relevant credentials have to be found simultaneously fresh at or after the request time.

$$
\begin{aligned}
&ForwardLooking(V_{DP}^{P,t_d}) \iff (\exists t\ t_{req} < t \leq t_d)(\forall c_i \in V_{DP}^{P,t_d})[(t_{req} < t^i_{ref,kmax(t)}) \\
&\wedge (\max_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{start,kmax(t)} \leq t^i_{ref,kmax(t)} < \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{end,kmax(t)}) \\
&\wedge Fresh(c_i, t^i_{ref,kmax(t)}) \wedge Fresh(c_i, t^i_{ref,kmax(t_d)}) \wedge Sat_{c_i}^{P,t} \wedge Sat_{c_i}^{P,t_d} \\
&\wedge \max_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{start,kmax(t_d)} < t_d < \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{end,kmax(t_d)}]
\end{aligned}
\tag{6}
$$

Property 1. There is a time interval during which all relevant credentials are simultaneously fresh after the request time.

Proof. Based on Eq. (6), all relevant credentials are simultaneously fresh during $[\max_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{start,kmax(t)}, \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{ref,kmax(t)})]$. Part of this interval is located after the request time since refresh has been done after it.

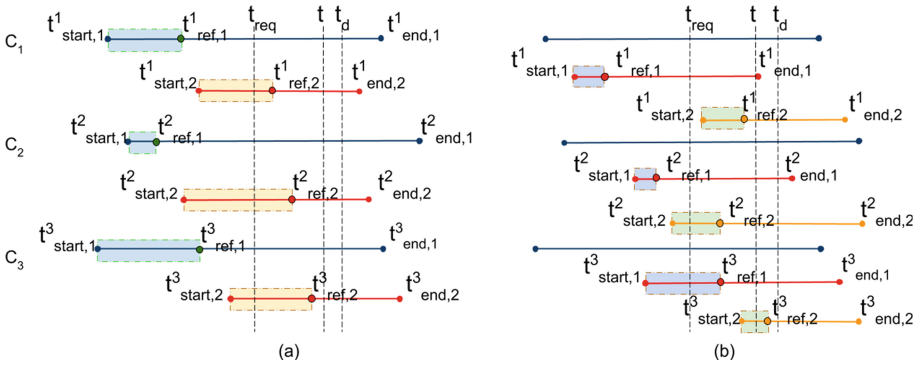

Fig. 5. Forward looking consistency

Property 2. Every forward-looking consistent view is interval consistent with request time as well.

Proof. Comparing Eqs. (5) and (6) shows forward-looking consistency is a restricted version of its preceding level, so the proof is trivial.

Property 3. Not every interval consistent with request time view is necessarily forward-looking as well.

Proof. At second level of consistency, only some credentials need to be refreshed after request time to compensate for lacking information. Whereas in forward-looking consistency, all have to be refreshed after the request time.

Comparing with Revocation-Based Scenario. Changing credentials in revocation scenario leads to hinder the access, whereas in refresh scenario, the `New-Value` in case of any changes would let the subject proceed. In Example 2, Bob's request to access project's documents at Jan 20th would be denied in a revocation-based scenario, however in refresh scenario access would be granted.

5 Limitations and Practical Issues

We presented three levels of consistency, where each higher level provides enhanced availability and safety at the cost of refreshing more frequently. We compared qualitative benefits of each level. Quantifying cost-benefit is highly implementation and application specific, and is beyond the scope of this paper. Furthermore, there are issues related to manage the risks inherent to applying ABAC in a distributed environment, since ABAC introduces new challenges in selecting appropriate trust models [9]. Finally, the formal correctness and appropriateness of the proposed criteria notwithstanding, the underlying information could be vulnerable to attack. The attack models would depend on the particular protocols and data structures used to implement credential transfer and refresh. As such they are out of scope for an abstract framework.

6 Conclusion

We formally characterize the safety and availability problem in multi-authority distributed ABAC systems. Our major contribution is to utilize the concept of refresh, which provides new attribute values rather than simply invalidating old ones. We propose three consistency levels which are totally ordered in strictness.

Acknowledgements. This work is partially supported by NSF CREST Grant HRD-1736209 and DoD ARL Grant W911NF-15-1-0518.

References

1. Adya, A.: Weak consistency: a generalized theory and optimistic implementations for distributed transactions. Ph.D. thesis, MIT (1999)
2. Bernstein, P.A., Goodman, N.: Concurrency control in distributed database systems. ACM Comput. Surv. **13**, 185–221 (1981)
3. Chase, M.: Multi-authority attribute based encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 515–534. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70936-7_28
4. Chase, M., Chow, S.S.: Improving privacy and security in multi-authority attribute-based encryption. In: ACM CCS (2009)
5. Cheminod, M., Durante, L., Valenza, F., Valenzano, A.: Toward attribute-based access control policy in industrial networked systems. In: IEEE WFCS (2018)
6. Garrison, W.C., et al.: On the practicality of cryptographically enforcing dynamic access control policies in the cloud. In: IEEE S&P (2016)
7. Goodrich, M.T., Shin, M., Tamassia, R., Winsborough, W.H.: Authenticated dictionaries for fresh attribute credentials. In: Nixon, P., Terzis, S. (eds.) iTrust 2003. LNCS, vol. 2692, pp. 332–347. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-44875-6_24
8. Harding, R., Van Aken, D., Pavlo, A., Stonebraker, M.: An evaluation of distributed concurrency control. Proc. VLDB Endow. **10**, 553–564 (2017)
9. Hu, V.C., et al.: Guide to attribute based access control (ABAC) definition and considerations. NIST SP 800-162 (2019)
10. Hu, V.C., Kuhn, D.R., Ferraiolo, D.F.: Access control for emerging distributed systems. IEEE Comput. **51**, 100–103 (2018)
11. Iskander, M.K., Wilkinson, D.W., Lee, A.J., Chrysanthis, P.K.: Enforcing policy and data consistency of cloud transactions. In: IEEE ICDCSW (2011)
12. Krishnan, R., Niu, J., Sandhu, R., Winsborough, W.H.: Stale-safe security properties for group-based secure information sharing. In: ACM FMSE (2008)
13. Krishnan, R., Sandhu, R.: Authorization policy specification and enforcement for group-centric secure information sharing. In: Jajodia, S., Mazumdar, C. (eds.) ICISS 2011. LNCS, vol. 7093, pp. 102–115. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25560-1_7
14. Lee, A.J., Minami, K., Winslett, M.: Lightweight consistency enforcement schemes for distributed proofs with hidden subtrees. In: ACM SACMAT (2007)
15. Lee, A.J., Winslett, M.: Safety and consistency in policy-based authorization systems. In: CCS. ACM (2006)
16. Lee, A.J., Winslett, M.: Enforcing safety and consistency constraints in policy-based authorization systems. In: TISSEC. ACM (2008)

17. Lupu, E.C., Sloman, M.: Conflicts in policy-based distributed systems management. IEEE Trans. Softw. Eng. **25**, 852–869 (1999)
18. Myers, M., Ankney, R., Malpani, A., Galperin, S., Adams, C.: X.509 internet public key infrastructure online certificate status protocol-OCSP (RFC 6960)
19. Park, J., Sandhu, R.: The UCON$_{ABC}$ usage control model. In: ACM TISSEC (2004)
20. Perrin, M.: Distributed Systems: Concurrency and Consistency. Elsevier, Amsterdam (2017)
21. Reyzin, L., Meshkov, D., Chepurnoy, A., Ivanov, S.: Improving authenticated dynamic dictionaries, with applications to cryptocurrencies. In: Kiayias, A. (ed.) FC 2017. LNCS, vol. 10322, pp. 376–392. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70972-7_21
22. Sciancalepore, S., et al.: On the design of a decentralized and multiauthority access control scheme in federated and cloud-assisted cyber-physical systems. IEEE IoT J. **5**, 5190–5204 (2018)
23. Shakarami, M., Sandhu, R.: Safety and consistency of subject attributes for attribute-based pre-authorization systems. In: NCS. Springer, Heidelberg (2019)
24. Van Steen, M., Tanenbaum, A.S.: Distributed Systems (2017)
25. Tamassia, R., et al.: Independently verifiable decentralized role-based delegation. IEEE Syst. Man Cybern.-Part A: Syst. Hum. **40**, 1206–1219 (2010)
26. Yang, K., Jia, X.: Attributed-based access control for multi-authority systems in cloud storage. In: IEEE ICDCS (2012)
27. Yang, K., Jia, X.: Expressive, efficient, and revocable data access control for multi-authority cloud storage. IEEE Parallel Distrib. Syst. **25**, 1735–1744 (2014)
28. Yang, K., et al.: DAC-MACS: effective data access control for multiauthority cloud storage systems. IEEE Inf. Forensics Secur. **8**, 1790–1801 (2013)
29. Zahoor, E., Ikram, A., Akhtar, S., Perrin, O.: Authorization policies specification and consistency management within multi-cloud environments. In: Gruschka, N. (ed.) NordSec 2018. LNCS, vol. 11252, pp. 272–288. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03638-6_17