An Efficient and Expressive Access Control Architecture for Content-based Networks

Joud Khoury, Samuel Nelson, Armando Caro, Vikas Kawadia, Dorene Ryder, and Tim Strayer
Advanced Networking, Raytheon BBN Technologies, Boston MA, USA
{jkhoury, snelson, acaro, vkawadia, dryder, tstrayer}@bbn.com

***Abstract*—Tactical content-based networks provide high military utility in dynamic mobile networks with intermittent connectivity and inherent disruption. Protecting the confidentiality of information exchanges (content and metadata) in such networks is particularly challenging since the publisher of information does not know who the subscribers are, yet the publisher wants fine-grained control over who has access to the information. Ciphertext Policy Attribute Based Encryption (CP-ABE) is a widely accepted cryptographic solution to this 1-many access control problem. This paper presents an efficient and expressive access control architecture for content-based networks based on CP-ABE. A key contribution of the paper is the efficiency of the proposed cryptographic solution which makes it practical in a resource constrained tactical network. We demonstrate our secure and efficient solution over a state-of-the-art tactical content based network, and we quantify its performance overhead.**

I. Introduction

A content-centric network provides efficient extensible information exchange between nodes that produce content (publishers) and those that consume it (subscribers) – the same node can act as a publisher node and a subscriber node at the same time. Publisher nodes describe the content of messages using *metadata*. Similarly, subscriber nodes query or register *subscriptions* in content and the content system delivers content to matching subscribers (queries or subscriptions). Content-based networks have several advantages, including (1) decoupled automatic discovery of content, (2) efficient interest-based delivery of information, and (3) simplified configuration especially under dynamic conditions.

Realizing the content-based networking paradigm in tactical mobile ad-hoc networks (MANETs) has high military utility yet it poses a unique set of challenges. These include rapidly and unpredictably varying links, constrained computing and networking resources, mobility, and potential network partitioning. CASCADE [1] is a state-of-the-art content-based network architecture for MANETs designed to overcome these challenges. By the intelligent replication and placement of content, CASCADE is able to serve relevant content to nodes in the MANET while operating in a resource-friendly fashion. Furthermore, CASCADE provides a rich ontology-based querying interface to quickly and intuitively discover content.

Protecting the confidentiality of published content and metadata while retaining the benefits of content-based systems is a particularly challenging task and is the main goal of this paper. Consider the following access control use cases for example: an explosive specialist detects and neutralizes an IED and publishes the information so that any marine could access it. A squad leader captures a high value target (HVT), publishes the image and location of the HVT such that a selected group can access it, and additionally publishes a highly sensitive piece of metadata about the HVT under a more restrictive policy (only squad leaders can access it). In the scenarios above, publishers are oblivious of subscribers yet still want fine-grained control over who can access their published information. The publisher of information encrypts using a policy defined over the access control attributes. Only users with attributes (and respective secret keys) that satisfy the policy are able to decrypt and access the information. Confidentiality is cryptographically enforced.

We extend the CASCADE architecture [1] to support access control with the following goals:

- Protect the *confidentiality of content and metadata*.
- Enable *fine grained and expressive access policies* to maximize the utility of the system.
- Provide publishers the *flexibility* to assign different access policies to different content items, and even to different metadata fields associated with the same content item.
- *Minimize the performance overhead* of access control on the overall system *throughput* and *latency* which is critical in a tactical setting.

To address these requirements, we exploit recent advances in functional encryption [2]. Specifically, for fine-grained access control, we resort to the secure and expressive Ciphertext-policy Attribute-based Encryption (CP-ABE) scheme by Waters [3] which we enhance for increased efficiency. The main contribution of this paper is a highly efficient CP-ABE construction and implementation that is demonstrated and quantified over a real-world content-based MANET, CASCADE [1]. We extend the CP-ABE construction of Waters [3] to support asymmetric bilinear maps. We additionally leverage the cryptographic micro-primitive optimizations made available by the Multi-precision Integer and Rational Arithmetic C/C++ Library (MIRACL) [4]. Together these enhancements significantly enhance the performance of our CP-ABE construction. Our overall access control solution is integrated into the Android-based CASCADE solution and is quantified over an internal 30 node testbed.

This material is based upon work supported by DARPA and SPAWAR under Contract No. N66001-12-C-4050. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA or SSC Pacific. Distribution Statement "A" (Approved for Public Release, Distribution Unlimited).

Related Work. CP-ABE is a widely accepted approach for securing information in loosely-coupled architectures. It has been proposed for confidentiality and privacy in a broad set of systems such as disruption tolerant networks (e.g, [5]), content centric networks (e.g, [6]), publish-subscribe systems (e.g, [7]), vehicular networks (e.g, [8]), etc. There is however little work on optimizing, and applying CP-ABE in a real world content-based MANET and quantifying its performance over Android mobile phones. We note that compared to the widely adopted construction in [9], [3] allows general representations of access structures based on linear secret sharing, is provable in the standard model [3], and can be made practically more efficient as we shall discuss.

The rest of the paper is organized as follows: section II briefly reviews the CP-ABE scheme. Section III presents the access control architecture and how it integrates with CASCADE's information dissemination protocols. Section IV evaluates the performance of the cryptographic primitives and the overall impact of access control on the system performance. Finally, we conclude and discuss future work in section V.

II. CP-ABE Background

We review the expressive CP-ABE scheme of Waters [3]. A ciphertext-policy attribute based encryption scheme consists of four algorithms: Setup, KeyGen, Encrypt, and Decrypt.

a) Setup(λ, U): takes the security parameter λ and attribute universe description U as input. We refer to the attribute universe as the *alphabet*. The algorithm outputs the public parameters PK and a master key MK.

b) KeyGen(MK, S): takes as input the master key MK and a set of attributes S that describe the key. It outputs a private key SK.

c) Encrypt(PK, M, $\mathbb{A}$): takes as input the public parameters PK, a message M, and an access structure/policy $\mathbb{A}$ over the universe of attributes. The algorithm will encrypt M and produce a ciphertext CT such that only a user that possesses a set of attributes that satisfies the access structure will be able to decrypt the message. We assume the ciphertext implicitly contains $\mathbb{A}$, i.e., the access policy is always in the clear.

We also restrict our attention to monotone access structures. Policies, or access structures, are expressive and may include threshold gates including **and** (n-of-n), **or** (1-of-n) or threshold **k-of-n** gates. The policy is monotone however so the **not** gate is not supported [3]. For example, using *KeyGen* a subscriber may be granted the following 2 attributes S ={`MARINES`, `SQUAD LEADER`} to form his secret key SK. A publisher encrypts a content item, using *Encrypt*, with the following boolean policy over the attributes: `SQUAD LEADER` **and** `MARINES`. The subscriber is able to decrypt the ciphertext since the subscriber attribute set satisfies the policy. Note that one may (inefficiently) support the **not** by explicitly including an attribute (say `X`) and its negative (`NOTX`) in the alphabet.

d) Decrypt(PK, CT, SK): takes as input the public parameters PK, a ciphertext CT, which contains an access policy $\mathbb{A}$, and a private key SK, which is a private key for a set S of attributes. If the set S of attributes satisfies the access structure $\mathbb{A}$ then the algorithm will decrypt the ciphertext and return a message M.

A key security property of CP-ABE is that it is *collusion-resistant*. This means that two nodes are not able to combine their attributes to decrypt a ciphertext unless at least one of them is able to decrypt on its own.

In terms of performance, the construction in [3] is for CP-ABE using symmetric elliptic curves. Our implementation, detailed in the Appendix, extends [3] to use asymmetric elliptic curves. Asymmetric curves are more practically and efficiently realizable for higher security levels [10]. Our CP-ABE implementation additionally uses several cryptographic primitive optimizations such as pre-computation, and multi-pairings which we discuss in more details along with the resulting performance improvements in section IV.

III. CASCADE Access Control Architecture

CASCADE is a content-based networking solution for MANET environments, led by Raytheon BBN Technologies as part of the DARPA CBMEN program [1]. At a high-level, CASCADE dynamically detects stable regions of the network, and pools the resources within those regions together to efficiently store, distribute, and serve content. These regions, referred to as *communities*, may be thought of as "mobile storage clouds" that automatically and efficiently organize all relevant content that they encounter using a Distributed Hash Table (DHT) per community. CASCADE prioritizes resource utilization by considering content metadata and community context (i.e., mission, role, location) to influence how content is moved and stored within and among communities. Applications search for content using rich semantic queries that are resolved to a set of unique Content Identifiers (CID) by a local Content Naming Subsystem (CNS).[1]

We discuss the access control architecture as it relates to CASCADE's main operations: *publish*, *query*, and *sync*.

A. Publish and Retag

Figure 1 shows a sketch of the publish operation steps.

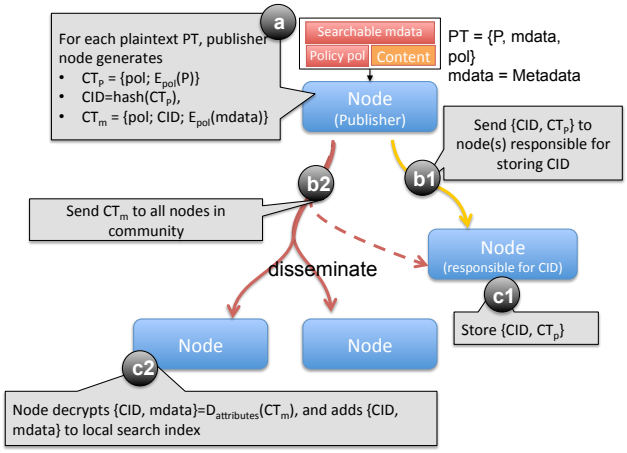

Fig. 1: CASCADE Publish Operation

a The publisher node wishes to publish a content P along with a set of searchable metadata mdata. The

[1]CNS was developed by a team led by Drexel Univeristy.

publisher wishes to control who can access the plaintext $PT = \{P, mdata\}$ using an access policy pol defined over the universe of access control attributes. The publisher creates two ciphertexts

- CT_P is the content ciphertext policy-encrypted using policy pol. The publisher hashes CT_P to create the content identifier CID. Note the ciphertext includes the policy in the clear as explained earlier in section II.
- CT_m is the metadata ciphertext policy-encrypted using policy pol. CID is included in the metadata ciphertext to reference the content that the metadata is associated with.

b1 Publisher node sends the content ciphertext CT_P to the node(s) responsible for storing the CID (depending on where CID falls in the DHT space).

b2 Publisher node disseminates the metadata ciphertext CT_m to all nodes in the community.

c1 Node(s) responsible for storing the content item receives and indexes the content ciphertext using CID.

c2 All nodes that receive metadata ciphertext CT_m, attempt to decrypt it for indexing purposes. Decryption succeeds only if a node possess the attributes (and corresponding private key) that satisfies encryption policy pol. Upon successful decryption, the node passes the metadata to the CNS so that the content is searchable. The node additionally stores CT_m used in the sync operation (described shortly).

Retagging is the process of adding metadata to a content item by a node possibly different than the original content publisher. For example, a node fetches a content item, decrypts the plaintext which was encrypted under policy pol, and decides to further annotate the item with a new set of metadata fields and publish those under a new policy. This process is similar to publish except the publication comprises solely metadata.

B. Query

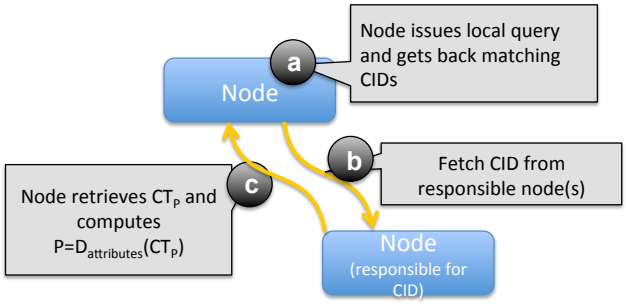

Fig. 2: CASCADE Query Operation

A sketch of the query operation steps is shown in Figure 2.

a A node issues a local query to the CNS and gets back a set of content identifiers CIDs. For each such CID, steps **b, c** below are repeated.

b Node issues a fetch request for CID from the node(s) responsible for storing the content.

c Node retrieves the content ciphertext CT_P and attempts to decrypt it using its private key. If the attributes satisfy the policy, the node successfully decrypts and consumes the content.

It is important to note that queries are satisfied locally by the CNS. Only metadata items that the local node has access to are indexed in the local CNS registry and used to satisfy queries.

C. Content Synchronization (sync)

When two communities get within proximity of each other, CASCADE provides a mechanism to synchronize their content. Syncing is the process of exchanging content and metadata items between nodes belonging to different communities so that items are available to both communities. There are three desired goals when two nodes sync, including

- *Correctness*: each of the nodes gets the content (and associated metadata) of the items that it is responsible for storing (e.g., those items that fall within the node's part of the DHT space)
- *Efficiency*: nodes should not exchange items they already possess and an item should not traverse the community more than once
- *Confidentiality*: content and metadata should only be accessible to nodes that are allowed to view it. Note that a node might store (and sync) content and metadata items it is responsible for even though it might not be able to access the items.

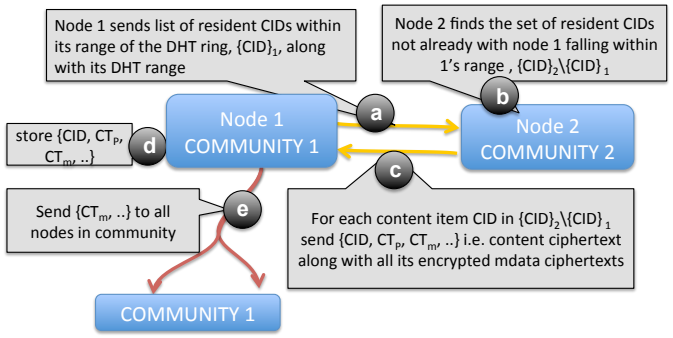

Fig. 3: CASCADE Sync Operation

Figure 3 shows two nodes, node 1 and node 2, who get within range and start the sync operation. Here, we only show how node 2 syncs its items with node 1 and it should be noted that the same process is performed in the opposite direction as well, and between all pairs of nodes in the two communities.

a Node 1 sends a list of its resident CIDs, call it $\{CID\}_1$, to node 2 along with a compact representation of the content it is responsible for.

b Node 2 computes the set of its resident CIDs that fall within node 1's subspace and are not yet at node 1, call this set $\{CID\}_2\backslash\{CID\}_1$.

c Node 2 sends the previously stored ciphertexts of all items in the set $\{CID\}_2\backslash\{CID\}_1$ including each item's content and all its metadata ciphertexts to node 1.

d Node 1 stores the item including encrypted content and metadata

e Node 1 then simply publishes the encrypted metadata just the same way a retag operation works (see section III-A).

During this sync, a community learns nothing about what the other community is interested in.

IV. Performance Evaluation

We first measure the performance of the CP-ABE cryptographic algorithms which we then use to directly deduce the impact of access control on the overall system.

A. CP-ABE Performance: Latency and Ciphertext Expansion

We first analyze the exact cost of the CP-ABE's cryptographic algorithms in terms of the number of low-level cryptographic primitives required for encryption and decryption (runtime performance). We discuss the performance optimizations used by our implementation mainly pre-computations, and multi-pairings and we show the resulting performance improvements. These performance improvements make our system more practical in a resource-constrained MANET.

1) Cryptographic Primitive Optimizations: In section II and the Appendix we presented our CP-ABE variant which uses prime order groups and an asymmetric bilinear map both of which result in significant speedups of the construction. To further enhance the performance of the cryptographic algorithms, we leverage the Multi-precision Integer and Rational Arithmetic C/C++ Library (MIRACL) [4] and the optimizations it provides (discussed next). The two key primitives for encryption and decryption are "exponentiations" and "pairings". MIRACL provides these optimized primitives which significantly speedup CP-ABE [4]. We discuss how the optimizations were applied in the Appendix.

Exponentiation with pre-computation: An exponentiation computes $g^x(mod\ n)$ where g is an element either of group $\mathbb{G}_1$ or $\mathbb{G}_2$. If g and n are known in advance, we can speed up the exponentiation by precomputing and storing a table of values $g^i(mod\ n)$ for different exponents i and replacing the exponentiation instead with lookups and multiplication [11]. This is directly applicable to CP-ABE encryption (and decryption) for example, since known fixed group elements in the public parameters (and the key) are raised to some exponent [3].

Pairing with pre-computation: A pairing computes the bilinear map $e(g_1, g_2)$ where $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ and $e(g_1, g_2) \in \mathbb{G}_T$ [3], [10]. If g_2 is fixed and is reused in multiple pairings, then it is possible to speed up the pairing using pre-computation of various parameters used in the Miller loop [12]. For example, CP-ABE *Decrypt* computes $e(C_i, L)$ and $e(D_i, K_x)$ where both L and K_x are fixed (for a given secret key) and C_i and D_i are different per encryption [3]. Pre-computation speeds up these pairings.

Multi-pairing (with pre-computation): A multi-pairing optimizes the computation of products of pairings [13]. These optimizations may be combined with pairing pre-computation if elements in $\mathbb{G}_2$ are fixed. For example, as described in the Appendix, CP-ABE *Decrypt* computes $\prod_{i \in I} e(C_i, L)\ e(D_i, K_{\rho(i)})$ where again the L and K_x are fixed for a given secret key. These computations can be significantly sped up using multi-pairing with pre-computation.

The core cryptographic primitives we shall utilize for implementing the CP-ABE algorithm are listed in Table I. Characterizing the performance of these primitives will directly characterize our system performance overhead. More specifically, the exact cost of a CP-ABE operation is a direct function of these primitives. All performance results hereafter assume a security level equivalent to AES 128 bits realized with the Barreto-Naehrig (BN) curve [14] and are measured on a single physical core on the Galaxy S4 Android phones running Android CynanogenMod 10.2 (ARM7 architecture, Qualcomm Snapdragon 600 chipset, 1.9 GHz quad-core CPU). From Table I, we see significant improvements as a result of the different optimizations. For example, using a multi-pairing with pre-computation on two elements speeds up a single pairing by around 75% (decryption time is sped up proportionally). Pre-computation also speeds up exponentiation in G_1 by around 70% (encryption time is sped up proportionally). Pre-computation speedups come at the cost of more caching as explained earlier. It is important to note that for all the numerical values in Table I, we additionally fixed the word size to 32 bits.

TABLE I: Performance of cryptographic primitives

Operation	Average time (ms)
Pairing $e(g_1, g_2)$	27.4
One more multi-pairing $e(g_1, g_2)e(g_1', g_2')$	11.2
Pairing pre-computation	4.7
Pairing with pre-computation	19.8
One more multi-pairing with pre-computation (ms)	6.9
Exponentiation (Exp) in G_1, g_1^s	1.13
Exp pre-computation in $\mathbb{G}_1$	18
Exp with pre-computation in $\mathbb{G}_1$	0.36
Exp in $\mathbb{G}_2$, g_2^s	2.53
Exp pre-computation in $\mathbb{G}_2$	31.1
Exp with pre-computation in $\mathbb{G}_2$	1.1
Hash to AES key in $\mathbb{G}_T$	1.36
Power in $\mathbb{G}_T$, $e(g_1, g_2)^s$	9
Power pre-computation in $\mathbb{G}_T$	99.3
Power with pre-computation $\mathbb{G}_T$	4.06

2) CP-ABE Performance: Based on the performance of the cryptographic primitives of Table I, Table II shows the exact cost analysis of the CP-ABE algorithm operations [3] described in section II and their performance speedups. For both Encrypt and Decrypt runtime operations, we show the "optimized", "actual", and "un-optimized" performance. The latter is the performance when pre-computation and multi-pairing optimizations are not applied. The optimized row indicates the target (expected) performance when the optimizations are applied based on Table II. The actual is the measured performance on the phones. Note that the gap between target and actual is mainly due to our Java implementation overhead which could be further optimized. We can clearly see up to 3x speedup relative to target and up to 2x speedups relative to actual in each of the encryption and decryption times as a result of the optimizations. This improvement directly translates to reducing the system latency overhead of access control proportionally as discussed next in section IV-B.

TABLE II: CPABE performance as function of number of attributes n in the access policy, and number of matching attributes L in the secret key

Operation	Cost	L=n=1	L=n=5	L=n=10
Encrypt: Exp w/ precomp G_1	n	0.4	2	4
Encrypt: Exp w/ precomp in G_2	$1+n$	2.26	6.8	12.47
Encrypt: Exp w/ precomp in G_T	1	4.5	4.5	4.5
Encrypt: hash-to-aes	1	1.46	1.46	1.46
Encrypt Total Time optimized (ms)		**8.67**	**14.8**	**22.47**
Encrypt Total Time actual (ms)		**9.77**	**26.73**	**38.8**
Encrypt Total Time un-optimized (ms)		**18**	**32.73**	**51.23**
Speedup Factor (actual)		1.84x	1.22x	1.32x
Speedup Factor (target)		2.1x	2.2x	2.3x
Decrypt: Exp w/ precomp in G_1	L	0.4	2	4
Decrypt: Exp w/ precomp in G_2	L	1.13	5.67	11.3
Decrypt: Multi-pairing w/ precomp	$2+L$	36.3	65.5	102
Decrypt: hash-to-aes	1	1.46	1.46	1.46
Decrypt Total Time optimized (ms)		**39.3**	**74.6**	**118.8**
Decrypt Total Time actual (ms)		**51.7**	**121.23**	**175.77**
Decrypt Total Time un-optimized (ms)		**82.87**	**201.26**	**349.27**
Speedup Factor (actual)		1.6x	1.6x	2x
Speedup Factor (target)		2.1x	2.7x	3x

TABLE III: CPABE ciphertext expansion as a function of number of attributes in the access policy n

Group Elements	Cost	n=1	n=5	n=10
number elements in G_1	n	64	320	640
number elements in G_2	$1+n$	256	768	1408
Total ciphertext expansion (bytes)		**320**	**1088**	**2048**

Ciphertext Expansion: Table III shows the CP-ABE ciphertext expansion/overhead required as a function of number of attributes n in the access policy. This is measured in terms of the number of group elements that comprise the ciphertext [3]. For the BN curve [14], an element in G_1 is 64 bytes whereas an element in G_2 is 128 bytes. Note that each group element comprises two coordinates (x, y) on the elliptic curve, where the size of each is equal to the size of the underlying field (256 bits for [14]). Accordingly, it is possible to reduce the ciphertext size (and accordingly the over-the-air bits) by half by representing each group element using only its x coordinate. This comes at a computation cost since the y coordinates will have to be computed on the fly during decryption.

B. System Latency

We measure the latency overhead of access control in CASCADE by measuring the reception latency when access control is enabled and when it's disabled and comparing the results. The reception latency is the end-to-end latency from when an item is published until it is received by any of the expected receivers. This was measured over our internal testbed comprising 30 static Android S4 phones connected to an EMANE network emulator and running the CASCADE stack along with an instrumentation application over emulated 802.11 WiFi links. The application published 875 content items over CASCADE of various sizes and expected 22200 total receptions (an item is received by multiple subscribers). Each such reception logs a latency data point. Figure 4 shows the latency cumulative distribution with and without access control. The $50th$ percentile latency with access control on is 0.94 sec compared to 0.7 sec when off i.e., access control adds around 240 ms of latency overhead. This cost grows linearly with the size n of the policy and the number L of satisfying attributes in the secret key.

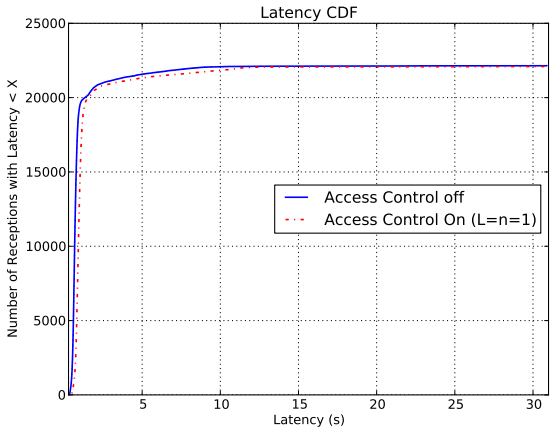

Fig. 4: Latency CDF comparison

As described earlier in section III-A, the publish operation requires two CP-ABE encryptions at the publisher node (one for metadata and one for content). These encryptions may run in parallel. In addition, each subscriber node requires a decryption for the metadata ciphertext and an additional decryption on the fetched content ciphertext (section III-B). This requires two encryptions and two decryptions to run serially increasing the end-to-end latency of the system accordingly. For the $L = n = 1$ attribute case shown in Figure 4, the expected latency overhead due to CP-ABE alone is around 123 ms per Table II (actual). The remaining 120 ms may be attributed to serialization delay due to ciphertext expansion and to the symmetric encryption/decryption and File I/O. Finally, note that our implementation is multi-threaded which means that CASCADE leverages additional cores to linearly increase the system throughput in the event that it is CPU-bound.

V. Conclusion and Future Work

This paper presents an efficient and expressive access control architecture for protecting the confidentiality of information exchanges (content and metadata) in content-based networks. We rely on Ciphertext Policy Attribute Based Encryption (CP-ABE) as the key enabling technology. Our CP-ABE implementation is made highly efficient by leveraging cryptographic and system optimizations. For 5 attributes, a CP-ABE encryption operation takes around 27 ms while a decryption takes around 120 ms. This is 2-3x improvement over an un-optimized counterpart.

In terms of future work, we plan to add key revocation and subscription privacy support. Bethencourt et al. [9] show how time-based key revocation would be possible using the current construction, and more recently Hur et al. [5] presented a more efficient construction. Protecting the *privacy* of subscriber interests requires no leakage about the subscriber's query to third parties. CASCADE protects privacy of interests to the most part simply because user queries are satisfied locally. However, when a user fetches the content from the responsible node(s) (Figure 2), the latter might learn what a specific user is interested in by examining the user's fetch requests over time. Recent system and cryptographic solutions for protecting subscriber privacy might be relevant here, see [7]

and references therein.

APPENDIX

An asymmetric CP-ABE scheme is presented extending the symmetric Setup, Encrypt, KeyGen and Decrypt algorithms of [3]. The extension is straightforward and with minor (but tedious) notational changes, the security proof of [3] applies to this asymmetric version since the hardness assumption used (Decisional Parallel Bilinear Diffie-Hellman Exponent Assumption) applies to both formulations. The security proofs are beyond the scope of this paper.

Setup: $(PK, MSK) \leftarrow Setup(U)$
The setup algorithm takes as input the number of attributes in the system U. It then chooses two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order p, generators $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ and U random group elements $h_1, \ldots, h_U \in \mathbb{G}_2$ that are associated with the U attributes in the system. In addition, it chooses random exponents $\alpha, a \in Z_p$. The public key is published as

$$\mathrm{PK} = \quad g_1,\ g_2,\ e(g_1, g_2)^{\alpha},\ g_2^{a},\ h_1, \ldots, h_U.$$

The authority sets $\mathrm{MSK} = (g_1^a, g_1^{\alpha})$ as the master secret key.

Encrypt: $Ciphertext \leftarrow Encrypt(\mathrm{PK}, (M, \rho), \mathcal{M})$
The algorithm takes as input the public parameters PK, the AES key used to encrypt the payload $\mathcal{M}$ and an LSSS access structure (M, ρ) where M is an $n \times \ell$ matrix and the function ρ associates each of the n rows of M to one of the U attributes.

The algorithm first chooses a random vector $\vec{v} = (s, y_2, ..., y_n) \in \mathbb{Z}_p^n$. These values will be used to share the encryption exponent s. For $i = 1$ to ℓ, it calculates $\lambda_i = \vec{v} \cdot M_i$, where M_i is the vector corresponding to the ith row of M. In addition, the algorithm chooses random $r_1, \ldots, r_\ell \in Z_p$. The ciphertext is published as $\mathrm{CT} =$

$$C = \mathcal{M} e(g_1, g_2)^{\alpha s},\ C' = g_2^{s},$$

$$(C_1 = g_2^{a\lambda_1} h_{\rho(1)}^{-r_1},\ D_1 = g_1^{r_1}), \ldots, (C_\ell = g_2^{a\lambda_\ell} h_{\rho(\ell)}^{-r_\ell},\ D_\ell = g_1^{r_\ell})$$

along with a description of (M, ρ). First, the size (number of rows) of the LSSS matrix is significantly reduced using the optimizations in [15]. In addition, all exponentiations use precomputations.

KeyGen: $SK \leftarrow KeyGen(\mathrm{MSK}, S)$
The key generation algorithm takes as input the master secret key and a set S of U attributes. The algorithm first chooses a random $t \in Z_p$. It creates the private key as

$$K = g_1^{\alpha} g_1^{at} \quad L = g_1^{t}, \quad K_j = h_j^{t},\ \forall j \in S$$

Decrypt: $M \leftarrow Decrypt(\mathrm{CT}, \mathrm{SK})$
The decryption algorithm takes as input a ciphertext CT for access structure (M, ρ) and a private key for a set of attributes S. Suppose that S satisfies the access structure and let $I \subset \{1, 2, \ldots, \ell\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, let $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ be a set of constants such that if $\{\lambda_i\}$ are valid shares of any secret s according to M, then $\sum_{i \in I} \omega_i \lambda_i = s$. (Note there could potentially be different ways of choosing the ω_i values to satisfy this.) The decryption algorithm first computes

$$e(C', K) / \left(\prod_{i \in I} (e(C_i, L) e(D_i, K_{\rho(i)}))^{\omega_i}\right) =$$
$$e(g_1, g_2)^{\alpha s} e(g_1, g_2)^{ast} / \left(\prod_{i \in I} e(g_1, g_2)^{ta\lambda_i \omega_i}\right) = e(g_1, g_2)^{\alpha s}$$

The decryption algorithm can then divide out this value from C and obtain the AES key $\mathcal{M}$. Following the approach in [16], the decryption algorithm can be reorganized to take advantage of the speed-ups due to precomputation and multipairing as follows. Let $A \in S$ be a minimal set of attributes that satisfy the access policy. First reduce the matrix M by removing rows associated with attributes that are not in A and remove redundant all-zero columns from the matrix. Next calculate the vector ω, which is the shares that reconstruct the secret (this is the first row of M^{-1}.) Set all $C_j \leftarrow \omega_j C_j$ and $D_j \leftarrow \omega_j D_j$. Where the same attribute is associated with more than one row of the M matrix, combine the associated C_j and D_j values by simply adding them. (This exploits bilinearity as $e(K_i, D_j).e(K_i, D_k) = e(K_i, D_j + D_k)$, which allows us to rewrite $D_i = D_j + D_k$). Finally recover the message as

$$\mathcal{M} = CT\ e(C_d, -K) e(\sum_{i \in A} C_i, L) \prod_{i \in A} e(K_i, D_i)$$

REFERENCES

[1] T. Strayer, V. Kawadia, A. Caro, S. Nelson, D. Ryder, C. Clark, K. Sadeghi, B. Tedesco, and O. DeRosa, "Cascade: Content access system for the combat-agile distributed environment," in *IEEE MILCOM*, 2013, pp. 1518–1523.

[2] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: a new vision for public-key cryptography," *Commun. ACM*, vol. 55, no. 11, pp. 56–64, Nov. 2012.

[3] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, effcient, and provably secure realization," Cryptology ePrint Archive, Report 2008/290, 2008, http://eprint.iacr.org/.

[4] Certivox, "MIRACL cryptographic SDK," http://www.certivox.com/miracl/.

[5] J. Hur and K. Kang, "Secure data retrieval for decentralized disruption-tolerant military networks," *IEEE/ACM Trans. Netw.*, vol. 22, no. 1, pp. 16–26, Feb. 2014.

[6] M. Ion, J. Zhang, and E. M. Schooler, "Toward content-centric privacy in icn: Attribute-based encryption and routing," in *Proceedings of ACM ICN '13*. New York, NY, USA: ACM, 2013, pp. 39–40.

[7] J. Khoury, G. Lauer, P. Pal, B. Thapa, and J. Loyall, "Efficient private publish-subscribe systems," In Proc. of ISORC'14, 2014.

[8] S. Ruj, A. Nayak, and I. Stojmenovic, "Improved access control mechanism in vehicular ad hoc networks," in *Ad-hoc, Mobile, and Wireless Networks*, ser. Lecture Notes in Computer Science, H. Frey, X. Li, and S. Ruehrup, Eds. Springer Berlin Heidelberg, 2011, vol. 6811, pp. 191–205.

[9] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. of the 2007 IEEE Symposium on Security and Privacy*, 2007, pp. 321–334.

[10] S. Galbraith, K. Paterson, and N. Smart, "Pairings for cryptographers," Cryptology ePrint Archive, Report 2006/165, 2006, http:eprint.iacr.org/.

[11] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 1996.

[12] C. Costello and D. Stebila, "Fixed argument pairings," in *Proc. of LATINCRYPT*. Springer-Verlag, 2010, pp. 92–108.

[13] R. Granger and N. Smart, "On computing products of pairings." Cryptology ePrint Archive, Report 2006/172, 2006, http:eprint.iacr.org/.

[14] P. S. L. M. Barreto and M. Naehrig, "Pairing-friendly elliptic curves of prime order," in *Selected Areas in Cryptography – SAC 2005*, ser. Lecture Notes in Computer Science, B. Preneel and S. Tavares, Eds., vol. 3897. Springer-Verlag Berlin/Heidelberg, 2006, pp. 319–331.

[15] Z. Liu and Z. Cao, "On efficiently transferring the linear secret-sharing scheme matrix in ciphertext-policy attribute-based encryption." Cryptology ePrint Archive, Report 2010/374, 2010, http://eprint.iacr.org/.

[16] J. Akinyele, C. Lehmann, M. Green, M. Pagano, Z. Peterson, and A. Rubin, "Self-protecting electronic medical records using attribute-based encryption." Cryptology ePrint Archive, Report 2010/565, 2010.