Secure Content Delivery in Information-Centric Networks: Design, Implementation, and Analyses

Satyajayant Misra
Computer Science
New Mexico State University
Las Cruces, NM, USA
misra@cs.nmsu.edu

Reza Tourani
Computer Science
New Mexico State University
Las Cruces, NM, USA
rtourani@cs.nmsu.edu

Nahid Ebrahimi Majd
Computer Science
New Mexico State University
Las Cruces, NM, USA
nmajd@cs.nmsu.edu

ABSTRACT

In this paper, we propose a novel secure content delivery framework, for an information-centric network, which will enable content providers (e.g., Netflix and Youtube) to securely disseminate their content to legitimate users via content distribution networks (CDNs) and Internet service providers (ISPs). Use of our framework will enable legitimate users to receive/consume encrypted content cached at a nearby router (CDN or ISP), even when the providers are offline. Our framework would slash system-downtime due to server outages, such as that recently experienced by Netflix, Pinterest, and Instagram users in the US (October 22, 2012). It will also help the providers utilize in-network caches for shaping content transmission and reducing delivery latency. We discuss the handling of security, access control, and system dynamics challenges and demonstrate the practicality of our framework by implementing it on a CCNx testbed.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Protocols; C.2.2 [**Network Security**]: Security Protocols

Keywords

ICN, CCN/NDN, content delivery, access control, security.

1. INTRODUCTION

According to the recent Cisco Visual Networking Index Forecast [3]: high bandwidth video makes 51% of the Internet traffic today and would rise to 54% by 2016; the sum of all video traffic would become approximately 86% of global traffic; and by 2014, mobile wireless devices will account for 61% of world Internet traffic. In the networking community, there has been a concerted push to redesign the Internet architecture to account for these future traffic trends. This push has culminated in the proposal of an *information-centric network* (ICN) approach, which has been at the core of several recent architectures (CCN/NDN [10], DONA [11], PSIRP [19], PURSUIT [8], and NetInf [4]). In this paper, we explore the design of a framework for highly available and efficient secure content delivery in such a network.

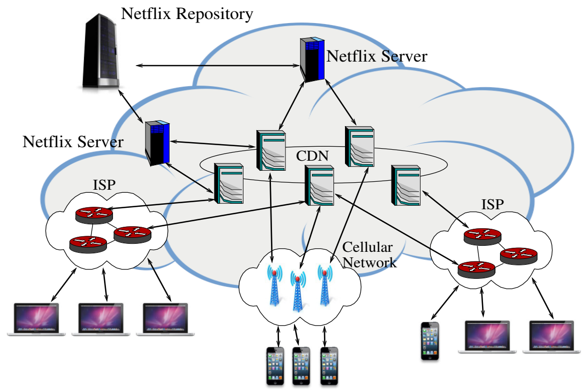


Figure 1: Internet content distribution architecture.

Motivation: Today, Content Providers (CPs), such as Netflix and Youtube, use content distribution networks (CDNs) to cache (store) content geographically closer to the users. The Internet hierarchy is composed of CPs at the top, followed by the CDNs (e.g., Akamai [1]), then the ISPs (e.g., Comcast, AT&T, Verizon), and culminating in the static/mobile end users, as shown in Figure 1. CDNs helps the CPs perform load balancing and reduce data latency, and also reduce redundant traffic requests in the network core. However, most CDN nodes are still at the edge of the ISPs (Fig. 1), thus the ISPs end up having to handle the explosive growth in data requests, especially repeated requests for the same (popular) data. In ICNs, decoupling of data from the source allows data to be stored *anywhere*. This in-network caching can be leveraged by ISPs in the information-centric Internet to reduce their network traffic load from redundant requests, and improve network scalability and data availability [22]. But, the important concern is, how do we ensure *high availability* of the cached data *only* to legitimate users?

Let's illustrate using Netflix as the CP and the CCN/NDN Internet architecture [10] for the ICN. Netflix uses CDNs, such as Akamai [1], to cache contents closer to the users. To ensure content access to only legitimate users, a user's Netflix player has to authenticate itself to a Netflix server hosted on a Cloud service (e.g., Amazon EC2). Upon authentication the player connects to a CDN node (based on proximity, network load, etc.) and obtains the content. The NDN architecture can now be leveraged to cache data at the ISPs' routers and serve the users' data needs.

However, if the cloud service is down, then the Netflix service is down–users cannot be authenticated! July 1 and October 22 of 2012 were the most recent occurrences (among many last year) of such an event resulting in Netflix (also Pinterest and Instagram) being unavailable for several hours [18].

In this scenario, data is available in routers close to the users, however, legitimate users are unable to authenticate themselves and access the data securely. In this paper, we wish to address this data-availability problem, which is widely applicable to all content delivery, by answering the question: Can we design a *practical security framework for ICNs* to deliver trusted content securely, efficiently, and with high availability, to legitimate users/subscribers? We present inferences from our initial investigation into designing such a security framework leveraging broadcast encryption and in-network caching. Our design is capable of handling $t > 0$ (order of $1 - 2$ million) revoked users efficiently, and can be augmented to handle more. The framework is applicable to all scenarios where server-based authentication is unavailable or difficult. We have also implemented our idea on a CCNx testbed and present results and analyses.

Section 2 presents the related work. Section 3 presents the system model and assumptions. We present our design details and protocol overviews in Section 4 and our experimental results and analyses in Section 5. In Section 6, we present our conclusion.

2. RELATED WORK

Several FIA architectures, such as CCN/NDN [10], DONA [11], PURSUIT and PSIRP [8], and NetInf [4] have been proposed in the literature. Caching and name-based addressing are integral components of all these architectures – the only two pre-requisites of our design. For our framework, we chose the CCN/NDN architecture for its simplicity and the availability of the CCNx [17] code-base. In CCN, content is split into packets/chunks, and each user sends a request (interest) containing the name of the chunk, into the network to obtain the chunk. This request propagates in the network toward the potential content provider. Any node in the path that contains the requested chunk can satisfy the interest. The data chunk follows the reverse path of the interest to reach the requester. Every intermediate node in the path has the chance of caching the chunk(s). NDN is composed of three main components, namely the *Forwarding Information Base* (FIB), the *Pending Interest Table* (PIT), and the *Content Store* (CS). The FIB table stores domains (name prefixes) and their corresponding outgoing face. The PIT table keeps track of outstanding interests; when an intermediate router forwards an interest packet through one of its outgoing faces, a corresponding 'pending interest' entry is created in the PIT. The last component, CS, is essentially a content repository.

Before presenting related work in BE, we note that secure group key communication [23] cannot be used here as it requires system-wide re-keying for each node revocation. Broadcast encryption (BE) was first proposed by Fiat and Naor [7] and has subsequently received a lot of attention [6, 15, 21]. In our framework, we choose to use the public key-based traitor-tracing t-resilient algorithm proposed by Tzeng and Tzeng in [21], which uses the Shamir's secret sharing algorithm as a building block. For this research, we propose to enhance the proposed BE algorithm for secure content delivery. Our aim is to design a framework to ensure that devices (even mobile devices) need only a few extra seconds, on account of the additional BE procedures, to securly obtain the key to decrypt the content.

We note that in this paper we only provide an overview of the cryptographic and protocol details, choosing to concentrate instead on the CCN/NDN implementation specific details and experimental testbed results for dissemination to the ICN research community. We will extend this paper with comprehensice details and analyses of the cryptography and the protocols only.

Broadcast encryption has found use in the real-world, for instance, AACS, HD DVD, and Blu-ray disc encryption. Digital Rights Management for DVD-video discs uses Content Scramable Systems, which is based on BE. However, those techniques do not apply automatically to Internet communication, especially for constrained mobile devices, which will make up the majority of the devices of the future.

3. MODELS AND ASSUMPTIONS

3.1 System Model

In this paper, we assume we have only one CP, say Netflix; our design scales to networks with multiple CPs. As mentioned in our motivating example in Section 1, we assume an obvious hierarchical set-up as shown in Fig. 1. The content flows from the CP, to the CDN nodes, to the ISP nodes, and finally to the end user who requests the content. It can be cached at any node in the path from the CP to the end user, thus enabling data-reuse.

For illustrating our design and experimentation, we assume the CCN/NDN [10] based Internet architecture. However, all ICN architectures share the same premise of caching and name based routing, and our design can be adapted to any ICN architecture.

In our design, we use the Schnorr group [14], where the large prime numbers Q and P are related as $P = rQ+1$ (i.e. $r = 2$), $n \leq (Q-t-1)$ is the number of users in the system, $t(<< n)$ is the revocation threshold, and all polynomial operations are performed in $\mathbb{Z}_Q^*$, the multiplicative group of integers of order Q. A first-time user registers with the CP to get its credentials and can obtain data from nearby sources (ISP routers or CDN nodes), subject to availability.

3.2 Set-up and Security Assumptions

We assume that the content is encrypted by the content provider, either at the servers or in the CDN (if it is trusted), using a popular symmetric key encryption algorithm. A content or a group of contents (set of movies) may be encrypted using the same secret key. Our objective is to ensure that the content is encrypted and cannot be used by an entity that is not a legitimate user/customer (not even the CDN or the ISPs). We achieve this by ensuring that only a legitimate user can obtain the symmetric key (transmitted using BE) to decrypt the content, whereas a fake or a revoked user cannot. We also assume that the user plays the content in a content provider specific player (e.g., Netflix, Hulu, etc.) and the player, which is non-tamperable, does not store the secret key used to decrypt the content. Without this assumption, no known encryption scheme can be used for security. A revoked user does not know the number of revoked users in the system or their identities. We assume that the user does not use tunneling or other location-cloaking mechanisms, such as the Tor network [20], to hide its network location.

3.3 Threat Model

In a set-up for content delivery, data security is of utmost importance. The use of symmetric key infrastructure, public key infrastructure, and our framework guarantees data security. However, there are several other attack scenarios. For instance, an attacker could mount denial of service

(DoS) attacks in the network by sending out a large number of interests or replaying interest/data packets. An adversary can pollute the cache of the routers in the network by sending out unpopular requests [24]. Traffic analysis attack can be performed on a specific user to identify her/his content access pattern [12]. A compromised or colluding user's keying materials can be extracted and used by an adversary, not part of the system, to mount impersonation and Sybil attacks. A few revoked users can collude to generate a key for an illegitimate user to decode the content. Also, there are standard attacks by an adversary, such as chosen plaintext attack (CPA), chosen ciphertext attack (CCA), and adaptive chosen ciphertext attack (A-CCA) [14, 21].

Our framework, operating in the CCN/NDN architecture, can address most of the aforementioned threats. For instance, the use of the sequence numbers in the interest and data packets, and caching at the edge routers can help neutralize replay attacks. Moreover, the use of the CCN/NDN interest packets make it difficult to mount DoS attacks on the system. Note that neither the NDN architecture nor our design require the users to identify themselves in the interest packets, thus ensuring privacy. Cache pollution attacks can be addressed in the NDN architecture satisfactorily [24]. The CPA, CCA, and A-CCA attacks can be handled by the BE protocol itself [21]. We discuss Sybil attack detection after discussing our framework.

4. DESIGN OF OUR FRAMEWORK

Our framework helps perform the following for content delivery in a CCN: (i) Allows ISPs to cache the content packets at their edge-routers, thus ensuring that the bandwidth is not wasted in redundant transmissions. (ii) Increases the availability of the content to the users by not requiring user authentication by an online server each time a content is requested. (iii) Ensures that only legitimate users can get access to the content and no revoked user can access the contents. In our design, all protocols are implemented at the top and the bottom level of the system hierarchy (Fig. 1). The BE algorithm is essentially a pluggable component for our framework. In this paper and in the experiments, we used a specific BE scheme proposed by Tzeng and Tzeng [21], which is a variant of Shamir's $(t+1)$-threshold secret sharing scheme. In the variant, the threshold $t+1$ of Shamir's scheme helps define a revocation threshold of t, which is the threshold for number of users that can be revoked from the system while still ensuring secrecy of the data. We assume that there are n legitimate users in the system and the number of revoked users ($|\mathcal{R}|$) can be at most t ($<< n$). We augment the proposed BE scheme to allow efficient encryption of the content, and ensure that the content can only be used by legitimate users, but not by the revoked users.

4.1 Design Overview of the Framework

Our framework consists of three major components: The first two components are performed at the server and are related to encrypting τ, the symmetric key for data encryption; the last component is formed at the client. In the **first component**, the server generates a polynomial of degree t and evaluates $n + t\,(>> t)$ number of points on it. The server distributes n of the evaluated points among the clients (one per client) and it keeps t of them as its own shares. In the **second component**, the server generates the enabling block, which contains the secret symmetric key (τ), and is used by the client in the last component for the secret extraction. The enabling block is forwarded to the routers, in which the contents are cached, and forms an integral part of the content. In the **third component**, a legitimate client extracts the embedded secret key from the enabling block that is downloaded along with the content, by using its share.

4.2 Basic Protocols

For illustration purposes, we use a server $\mathcal{S}$ to illustrate the computations at the server(s) or the CP and u_j to illustrate the end-user. Now we sketch the protocols that make up the three steps, note that the u_j needs to be registered to obtain its share.

4.2.1 Polynomial and Shares Generation

This protocol is performed at the server and takes as input a prime number Q, $ZQrand()$ a random number generator in the field of Q, Z_Q^* the multiplicative group of integers of order Q, the number of users $n < Q$, and the maximum number of allowed revoked users t. The server calculates random coefficients $a_0, \ldots, a_t$ of a t-degree polynomial $p_t(x)$. Then it calculates each user u_j's share/tuple $T_j =< x_j, f(x_j) >$, where x_j is generated randomly and $f(x_j)$ is calculated, both in the field Z_Q^*. The server calculates t-tuples that it stores as the server share E.

For registration, a first-time user creates a verified user profile. On successful access to the system, the server transmits to the user u_j, T_j encrypted using u_j's public key P_j, the signature, and the timeout TO so that u_j can correctly extract the secret key and then decrypt the content.

4.2.2 Generation and Encryption of Enabling Block

The server (or the CP) encrypts the content using a symmetric key algorithm and a secret key τ. For simplicity of exposition, we assume that $\tau \in \mathbb{Z}_Q^*$, where Q is of the order of the number of users (in millions). A bigger key, say 128-bit key for AES, can be handled by splitting the key $\Upsilon = \{\tau_1||\ldots||\tau_b||\ldots||\tau_m\}$, where each $\tau_b \in \mathbb{Z}_Q^*$. This protocol, implemented at the server, deals with the generation of the enabling block and its secure transmission as a part of the encrypted content. An enabling block is an integral part of any BE scheme. It contains information for a legitimate user to extract the secret key τ and is delivered to the user along with the data packets.

This protocol takes the server share E, a random number $r \in \mathbb{Z}_Q^*$, a generator g of a cyclic group $\mathbb{G}_Q$ of order Q, coefficient a_0, and τ. The server calculates $\gamma = \tau \cdot g^{ra_0}$ (γ is the encrypted symmetric key) and creates the enabling block for the user to decrypt γ and obtain τ. The user can decrypt γ using the enabling block and by the Lagrangian interpolation method [21]. However, the Lagrangian interpolation method requires computations of t Lagrangian coefficients at the client (user) with running time complexity of $\mathcal{O}(t^2)$, which can be excessive for large values of t. Thus, we have devised a mechanism for pre-computing the Lagrangian coefficients at the server and sending them as a part of the enabling block to reduce the running time complexity of the computation at the user to $\mathcal{O}(t)$. *This precomputation, is our novel contribution to the state-of-the-art in BE, making it usable in mobile devices.* This protocol requires $\mathcal{O}(t)$ modular exponentiations, which make the bulk of the running time. This results in decryption at the user taking around 4 seconds for $t = 1$ million. Additionally, as we will show us-

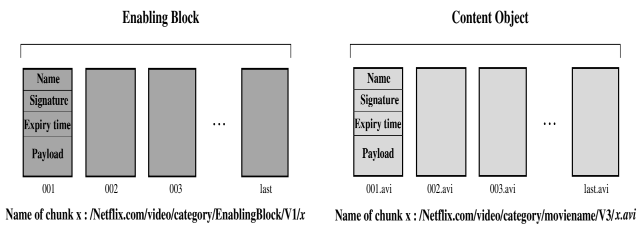


Figure 2: Naming scheme for Chunks.

ing our experiments the increase in the size of the enabling block due to the addition of the coefficients is acceptable. The server signs the enabling block using it's private key to guarantee provenance.

4.2.3 Secret Extraction at the User

The user u_j obtains the enabling block from a router in it's neighborhood, verifies the source of the enabling block by verifying the signature. On successful verification it computes the t complete Lagrangian coefficients (from the partials) using its share and also the Lagrangian coefficient corresponding to it's own share, resulting in a total of $t+1$ coefficients. The user u_j uses the computed Lagrangian coefficients to extract the secret key τ from γ using the Lagrangian interpolation method. This protocol requires $\mathcal{O}(t)$ modular exponentiations, which again make the bulk of the running time. We do not go into the details of the protocol, but refer the reader to [21] for more details. We also remind the reader that our framework can use any BE algorithm.

4.3 CCN/NDN Architecture Specific Details

User Registration: In the CCN/NDN architecture all communication are initiated with interests. In our approach, for registration a user u_j sends a *registration* interest to the CP. The format used for the name of the registration interest is: $/Netflix/Registration/Unique\_User\_ID$. This interest packet contains the u_j's credentials, encrypted with the CP's public key and signed by u_j's private key. The CP then replies to u_j with a data packet containing u_j's unique valid share encrypted with u_j's public key. This exchange is a unicast exchange between the CP (or its server) and the client u_j and no router on the path caches the reply. Even if the data is cached by a malicious router it does not undermine communication secrecy.

Chunk Creation: After the registration procedure, the user can send *data* interest packets. In the CCN/NDN architecture one single interest elicits one data packet, thus a large content has to be broken down into smaller packets (chunks). In our framework, the content is split into smaller chunks with a unique name per chunk. The size of each chunk is a constant, in our experiments we used chunk-size of 4kB, but chunk sizes may conform to lower layer packet sizes (e.g., a 1300 byte packet conforms to the Ethernet as the Link layer protocol and prevents fragmentation). Fig. 2 illustrates the splitting of the content – the enabling block and the content are both split into equal sized chunks and given appropriate names to distinguish them.

Packet Naming: In CCN/NDN a user needs to know the name of each individual data chunk to generate the corresponding interest packet, thus a naming scheme is important. Each name has to be unique and a complete ordering of the chunks can be obtained using the names. As shown in Fig. 2, we choose a name in accordance with the hierarchical naming convention of CCN/NDN – the first segment of the name is the CP's name, the second is the fact that it is a video, the third is the category (Sci-Fi, Comedy, etc.), the fourth is the type of data (Enabling block or movie name), the fifth is the version, and the last part of the name represents the number of the chunk. We assume that movies that belong to a particular category are all encrypted using the same key and hence can reuse the same enabling block. We propose two schemes to assign the chunk-number, namely sequential numbering and random numbering, each with inherent advantages.

Sequential Numbering: With reference to Fig. 2, in sequential naming, each chunk of the enabling block has a name {`/Netflix.com/video/category/EnablingBlock/V1/x`} , where the chunk number $x \in \{001, 002, \ldots\}$. In this set-up, the client can easily generate the name for the first interest packet if it knows the naming convention and then increments the x value for each subsequent packet. We assume that the first chunks of the enabling block and the content object contain the x for their corresponding last chunks so that the user knows when to stop. The sequential scheme is easy to implement, but enables an attacker to probe the cache at the router/proxy to perform traffic analysis [12].

Random Numbering: In the random numbering, the value of x for the first packet is 1, however, each subsequent packet has a random x value. Each chunk carries the x value of the next chunk so that the user can construct the name of the next chunk. The use of random sequence numbers can undermine the traffic analysis attack mentioned above, however, this approach may also undermine the ability to aggregate request for several chunks in one interest packet. We are exploring enhancements in this direction.

Versioning: The names of the enabling block and the movie both contain version information ($V1$ and $V3$ respectively in Fig. 2). The content may be available in several qualities, each will have its own version number. Versioning of the enabling block is essential to handle the dissemination necessitated by user revocation. A router can use the version number to distinguish between two versions of the enabling block and replace the old version with the new one, even before the old one times out.

User Revocation: A user can cancel his subscription, effectively revoking his content access privilege. The user requests service-cancellation using a *revocation* interest packet. The *revocation* interest is sent out to the CP in the same way as the registration request. This request contains the information that the user provides at the registration time in addition to the revocation part. The CP, validates the revocation request according to the provided credentials and the user's signature. After that the CP regenerates the new enabling block according to the procedure mentioned in section 4.2. A revoked user has access to the enabling block and hence the encrypted content cached in the router, until the enabling block is updated. Hence the updated enabling block has to be disseminated in the network. This may be done in a proactive manner (immediately after a revocation), periodically (every week or month), or by a pull-mechanism from the network (when the enabling block at routers times out, they seek new versions of the enabling block). The periodic or the pull-mechanism are obviously better than the system-wide dissemination.

4.4 System Dynamics: A Discussion

There are several factors that lend dynamism to the system. For instance, *(i)* how can we handle the need to revoke a subscribed user at the end of his subscription? *(ii)* Can we handle the case where the number of revoked users is more than the threshold t? *(iii)* How do we handle a new user when the system reaches its user capacity n? We present some initial attempts to answer the questions.

When a user u_r has to be revoked in a secret sharing based scheme as the one used in our framework, the server replaces one of its t tuples in the server shares, which forms the part of the enabling block, with the tuple T_r of u_r. This disables u_r from decrypting τ. Then the updated enabling block is disseminated in the network.

The scenario of number of revoked users ($|\mathcal{R}|$) being more than the threshold t can be handled in a preemptive or a reactive manner. In the preemptive approach, the system can be re-keyed by the CP when $|\mathcal{R}|$ gets close to t. In the reactive approach, the key τ can be broken into several sub-keys and each sub-key can be encrypted independently. The server shares for each of the corresponding enabling blocks will be chosen as ${}^{|\mathcal{R}|}C_t$ ($|\mathcal{R}|$ choose t), such that a revoked user's share appears in at least one of the enabling block. This ensures that the revoked user cannot accurately decrypt τ.

When the system reaches user capacity, there is some scope for reuse of the tuples, which the revoked users' tuples replaced in the server's share. However, in the worst case, there is a need for reinitializing the whole system including distribution of the new tuples to the users and the corresponding enabling block(s).

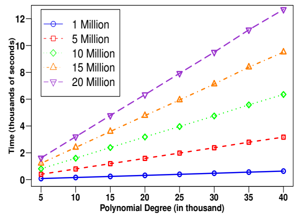
4.5 Security Analysis

The outstanding security concerns for our framework include Sybil attacks, collusion attacks, and well-known attacks, such as CPA, CCA, and A-CCA. With the use of the NDN architecture, even if a client signs the interest packet, there is no way to stop a Sybil attack in the network as a Sybil node possesses the keys of the legitimate node. This reconfirms Douceur's [5] observation of the difficulty of containing Sybil attack without a central verification entity. However, one possible way to identify a Sybil node is through mandated periodic user-credential verification by the CP or a proximal router. The verification can use gross location information (based on location of closest ISP router or CDN node) to obtain a gross estimate of the user's geographic location. If a user appears at multiple locations simultaneously (or over a short span of time) it would imply a Sybil attack, and the user can be revoked.

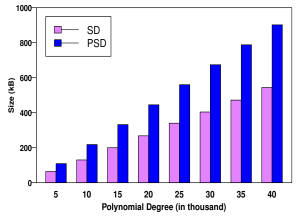
A set of colluding nodes can create a new share for a new malicious (illegitimate) node, however this requires at least $t+1$ malicious/revoked nodes to collude, armed with the knowledge of the prime numbers used to bootstrap the system, so that they can re-generate the polynomial using their shares. With $t+1$ being of the order of thousands (or millions) that is unlikely. For addressing the other attacks, such as CPA, CCA, and A-CCA, we refer the readers to [21] – the proofs are similar and we omit them here for brevity while stating that the secrecy of τ is preserved against those attacks. To counter longstanding threats of content piracy, τ can be renewed at regular intervals and content re-encrypted. Several user privacy threats have been identified in the ICN literature [2, 12]. We do not discuss them as they are outside the scope of this paper.

5. IMPLEMENTATION RESULTS IN CCNX

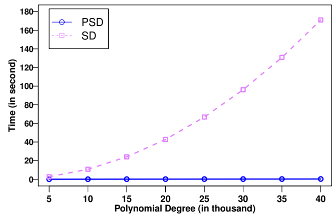
We implemented our framework in our CCNx-0.7 [17] testbed, which currently consists of three nodes. Each node



(a) Polynomial and Users Share Generation time



(b) Enabling Block Size



(c) Symmetric Key Extraction

Figure 3: Results from Protocols Implementation: (a) Time taken to generate $p_t(x)$ and the users' shares; (b) Enabling Block Size: with precomputation (PSD) and without precomputation (SD); (c) Time required for secret key extraction.

in the testbed has an Intel Core i7 processor with an 8 GB total system memory and each core clocked at 2.4 GHz. Only one core was used in the experiments. The protocols were implemented in C++ using the gcc compiler version 4.5.2 and used the GNU Multi-Precision Arithmetic (GMP) library [9] for cryptographic operations. Our multimedia content – a 24.1 MB video – was hosted (using the *ccnputfile* command) in the content store of one testbed node (server). The rest of the nodes were clients, which sent out the interests to the server using the *ccnsimplecat* command to receive the corresponding chunks. We split the content into chunks of 4 Kilobytes each.

We implemented the Polynomial Generation protocol, the Enabling Block Generation and Encryption protocol, and the Secret Extraction protocol. In our implementation, the total number of users ranged from 1M to 20M (close to Netflix's user base size) in multiples of five, and the value of t ranged from 5K to 40K in increments of 5K, where M and K stand for million and thousand respectively. In our framework, the server performs pre-computation of the Lagrangian coefficients. We compare our framework (denoted as *PSD*) with a replica of our framework, without the server-side pre-computation (denoted as *SD*). Our results were averaged over 100 experiment runs.

Fig. 3 displays some of our results. Fig. 3(a) shows the time taken by the server to generate a polynomial ($p_t(x)$) of a certain degree, including generating its random coefficients ($\{a_0, \ldots, a_t\}$), and then evaluating $p_t(x)$ at $n+t$ points. The average time for the procedure for t ranging from 5K-40K and n ranging from 1M to 20M is shown here. The X-axis represents different values for t and the Y-axis represents the time in thousands of seconds. The least running time corresponds to $p_t(x)$ with the polynomial degree being $t=$ 5K, and 1M users (number of points on the polynomial is 1.005M), requiring 80 seconds. The highest running time corresponds to $t=40K$ with 20M users and takes about 12687 seconds. *The polynomial generation procedure is the most time consuming component of our framework, however, it is executed by the server only and can be performed offline.*

It is easy to see why the running time is dependent on the value of t and n and why it scales linearly – the generation time for 20M users is 20 times more than that for 1M users, for the same values of t.

Fig. 3(b) shows the size of the enabling block in the PSD and SD cases. The X-axis represents the polynomial degree (t) and the Y-axis represents size in KiloBytes. The size of the enabling block is independent of the number of members, is proportional to the revocation threshold t, and increases linearly with an increase in t. It is desirable to have an enabling block of small size, since the server has to generate a new enabling block each time a user is revoked. Also, with service differentiation (basic, premium, etc.), we may have multiple enabling blocks need to be stored at each router in the network, corresponding to different sets of users and possibly for different set of contents. Hence, smaller the size of the enabling block lesser the network overhead.

We note that the extra server-side precomputation results in the enabling block in PSD being larger than that in SD ($> 50\%$ for $t = 40$K). However, the corresponding reduction in extraction time at the client is significantly lower (refer Fig. 3(c)). In the worst case, the size of the enabling block in PSD is around 900kB for $t = 40$K. A standard two-hour Netflix movie for mobile devices has a size of around 300 MB [13]; the enabling block adds less than 0.3% overhead.

Fig. 3(c) presents a comparative analysis of the secret key extraction time in PSD and SD. The key extraction time in SD grows super-linearly with increasing t, in comparison, PSD is impressive, needing less than one second for all values of t. To demonstrate the scalability of our approach, we obtained statistics for higher values of t as well ($t \in \{0.1\text{M}, 0.3\text{M}, 0.5\text{M}, 0.7\text{M}, 1.0\text{M}\}$). Even when t is 1 million, the size of the enabling block is 24.2 MB (8% of a standard movie [16]) and it takes 4.17 seconds (0.06% of the movie time) to extract τ using one 2.4 GHz processor. For perspective, the Samsung Galaxy Note 2, a possible end-device, has 2 GB RAM and a 1.6 GHz quad-core processor.

Revocation threshold of 1 million is a large number as can be seen from recent Netflix statistics [16] and may happen over a long time period. This would allow time for refreshing the enabling blocks to handle user revocation.

6. CONCLUSIONS

In this paper, we present an efficient framework for secure and high availability content delivery in ICNs. We sketch the protocols, present the architecture details and experimental results demonstrating the framework's practicality. In the future, we will study in-depth application of our framework to other ICN architectures and optimize and implement our protocols on smartphones and a larger testbed.

7. REFERENCES

[1] Akamai. http://www.akamai.com/.

[2] A. Chaabane, E. De Cristofaro, M. Kaafar, and E. Uzun. Privacy in content-oriented networking: Threats and countermeasures. *arXiv:1211.5183*, 2012.

[3] Cisco. Cisco visual networking index, 2012. `http://www.cisco.com/en/US/netsol/ns827/networking_solutions_sub_solution.html`.

[4] C. Dannewitz. NetInf: An information-centric design for the future Internet. In *3rd GI/ITG KuVS Workshop on The Future Internet*, 2009.

[5] J. Douceur. The sybil attack. *Peer-to-peer Systems*, pages 251–260, 2002.

[6] Y. Duan and J. Canny. Scalable secure bidirectional group communication. In *IEEE INFOCOM*, pages 875–883. IEEE, 2007.

[7] A. Fiat and M. Naor. Broadcast encryption. In *CRYPTO*, pages 480–491, 1994.

[8] N. Fotiou, P. Nikander, D. Trossen, and G.C. Polyzos. Developing information networking further: From PSIRP to PURSUIT. In *Proc. 7th International ICST Conference on Broadband Communications, Networks, and Systems*, pages 1–13, 2010.

[9] The GMP Library, 2012. `http://www.gmplib.org`.

[10] V. Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N.H. Briggs, and R.L. Braynard. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 1–12. ACM, 2009.

[11] T. Koponen, M. Chawla, B.G. Chun, and *et al.* A data-oriented (and beyond) network architecture. *ACM SIGCOMM CCR*, 37(4):181–192, 2007.

[12] T. Lauinger, N. Laoutaris, P. Rodriguez, and *et al.* Privacy implications of ubiquitous caching in named data networking architectures. Technical report, TR-iSecLab-0812-001, iSecLab, 2012.

[13] App Makers Worry as Data Plans Are Capped, June 6, 2010. `http://www.nytimes.com/2010/06/07/technology/07data.html?_r=0`.

[14] A.J. Menezes, P.C. Van Oorschot, and S.A. Vanstone. *Handbook of applied cryptography.* CRC, 1997.

[15] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *CRYPTO*, pages 41–62, 2001.

[16] Netflix: 10 years in 3 charts, October 25, 2011. `http://www.splatf.com/2011/10/netflix-10-years/`.

[17] PARC. Ccnx. `http://www.ccnx.org/`.

[18] Storm Crushes ... Netflix, Pinterest, Instagram, 2012. `http://www.wired.com/wiredenterprise/2012/06/real-clouds-crush-amazon/`.

[19] S. Tarkoma, M. Ain, and K. Visala. The publish/subscribe internet routing paradigm (psirp): Designing the future internet architecture. *Towards the Future Internet*, page 102, 2009.

[20] Tor Project: Anonymity Online. `http://www.torproject.org/`.

[21] W. Tzeng and Z. Tzeng. A public-key traitor tracing scheme with revocation using dynamic shares. In *Public Key Cryptography*, pages 207–224, 2001.

[22] S. Wang, J. Bi, J. Wu, Z. Li, W. Zhang, and X. Yang. Could in-network caching benefit information-centric networking? In *7th Asian Internet Engineering Conference*, pages 112–115, 2011.

[23] C.K. Wong, M. Gouda, and S.S. Lam. Secure group communications using key graphs. *IEEE/ACM Transactions on Networking*, 8(1):16–30, 2000.

[24] M. Xie, I. Widjaja, and H. Wang. Enhancing cache robustness for content-centric networking. In *IEEE INFOCOM*, pages 2426–2434. IEEE, 2012.