ACCEPTED FROM OPEN CALL

Security and Privacy for Mobile Edge Caching: Challenges and Solutions

Jianbing Ni, Kuan Zhang, Athanasios V. Vasilakos

Abstract

Mobile edge caching is a promising technology for next-generation mobile networks to effectively offer service environments and cloud-storage capabilities at the edge of networks. By exploiting the storage and computing resources at the network edge, mobile edge caching can significantly reduce service latency, decrease network load, and improve the user experience. On the other hand, edge caching is subject to a number of threats regarding privacy violations and security breaches. In this article, we first introduce the architecture of mobile edge caching, and address the key problems regarding why, where, what, and how to cache. Then we examine the potential cyber threats, including cache poisoning attacks, cache pollution attacks, cache side-channel attacks, and cache deception attacks, which result in huge concerns about privacy, security, and trust in content placement, content delivery, and content usage for mobile users, respectively. After that, we propose a service-oriented and location-based efficient key distribution protocol (SOLEK) as an example in response to efficient and secure content delivery in mobile edge caching. Finally, we discuss the potential techniques for privacy-preserving content placement, efficient and secure content delivery, and trustful content usage, which are expected to draw more attention and efforts into secure edge caching.

Introduction

Due to the vastly increased number of connected devices in the Internet of Things (IoT) [1], wireless networks are expected to provide supremely high data rates and extremely low latency to enable the emerging applications, such as remote control of smart vehicles, industrial automation, and virtual/augmented reality. By integrating the disruptive technologies, mobile devices have ultra-reliable and affordable network access to satisfy the quality of experience in wireless environments, thereby making our life more convenient. As one of the key technologies, multi-access edge computing [2] is becoming increasingly popular, offering a service environment and cloud-computing capabilities in an effective manner at the edge of wireless networks. By leveraging the computing and storage resources on the edge nodes (e.g., macro base stations, WiFi access points, network switches, video cameras, and roadside infrastructure), popular content can be temporarily maintained on the edge nodes to respond to frequent access requests from user equipment, which is referred to as mobile edge caching [3]. Since the contents are delivered directly from edge nodes instead of from the remote cloud if they are cached on edge nodes when requested, network edge caching can significantly reduce the content delivery latency, alleviate the backhaul traffic, and improve spectrum efficiency and energy efficiency [4].

Despite the appealing benefits of the quality of service, caching at the network edge has been subject to a number of controversies regarding security breaches and privacy violations. For example, Denial-of-Service (DoS) attacks and wireless jamming can consume the bandwidth and computing resources at the network edge. The open edge network creates the real-world attack surface for adversaries that aim to compromise or control the edge caches. The attackers might acquire the stored data at the edge caches that are located in their vicinity, as well as the data that are from other locations in distributed edge storage, such that the sensitive information can be extracted because of the context awareness. In addition, the edge network providers might be curious about the content stored on their caches and utilize the inference attacks to identify the hidden information about mobile users, such as trajectory and personal identifiable information. However, with limited computation, power, and memory, complex secure and privacy-preserving mechanisms cannot be straightforwardly deployed on the edge caches. Therefore, it is of paramount importance to explore a deeper understanding of the potential attacks and design practical real-world solutions for reasoning about the security and privacy of mobile edge caching.

In this article, we aim to examine security and privacy threats in mobile edge caching, and explore new models, techniques, and approaches to reason about the security, privacy, and trust of mobile edge caching. Specifically, we start with the architecture of mobile edge caching and address the key issues about edge caching, that is, why to cache, where to cache, what to cache, and how to cache. We then examine a series of threats on security and privacy that are grouped into two categories: generic threats, such as DoS attacks, wireless jamming, malware attacks, and man-in-the-middle attacks; and cache-specific threats, including cache poisoning attacks, cache pollution attacks, cache side-channel attacks, and cache deception attacks. However, the research about edge caching security is still in the early stage. We

Digital Object Identifier:
10.1109/MWC.001.2000329

Jianbing Ni is with Queen's University; Kuan Zhang is with the University of Nebraska-Lincoln; Athanasios V. Vasilakos is with the University of Technology Sydney.

1536-1284/20/$25.00 © 2020 IEEE

discuss the privacy, security, and trust challenges in mobile edge caching, and explore the potential solutions to preserve privacy in content placement, security in content delivery, and trust in content usage. To identify the approaches of securing content delivery, we propose a Service-Oriented and Location-based Efficient Key distribution protocol (SOLEK) for secure content sharing based on certificateless proxy re-encryption as an example. Finally, we discuss future research directions of privacy-preserving content placement, efficient and secure content delivery, and trustful content usage, to draw more efforts for securing edge caching.

Overview of Network Edge Caching

In this section, we present the general architecture of mobile edge caching. As shown in Fig. 1, mobile edge caching involves several entities, that is, user equipment, access networks, mobile core network, and the Internet. User equipment (UE), such as smart phones, smart vehicles, smart meters, and smart home produces, carried by mobile users or deployed in the ground, are capable of accessing different services in the Internet through access networks and mobile core networks. In access networks, different wireless communication technologies, including radio access network, WiFi, small cells, and 5G new radio, are leveraged to build communication channels for UE with the support of infrastructure, such as base stations, edge routers, and switches. Mobile core network is the heart and acts as an anchor point for multi-access technologies. 5G core comprises pure, virtualized, software-based network functions or services, and can be instantiated within multi-access edge-cloud computing infrastructures. The Internet is the data network that provides a large variety of services to the connecting mobile users.

The problems of why, where, what, and how to cache are the key issues in mobile edge caching. Why to cache refers to the advantages of mobile edge caching regarding the quality of experience of mobile users. Where to cache refers to location selection for caching content. What to cache refers to the caching content that has short lifetimes and consists of different types, such as documents, videos, radios, and IoT data. How to cache refers to caching policy selection and caching scheme design.

Why to Cache?

The primary reason for caching is to make data access less expensive on time, bandwidth, and resource costs.

Service Delay Reduction: By storing content close to mobile users, the service delay can be significantly reduced. This feature is quite attractive for data delivery of latency-sensitive applications, such as video streaming that will account for 78 percent of all mobile traffic by 2021.

Traffic Burden Releasing: As duplicate content causes severe traffic burden over hackhaul links, the traffic can be reduced up to 35 percent by storing the requested content at the network edge [2]. Caching in 3G and 4G LTE networks can reduce mobile traffic up to two thirds [5].

Energy and Spectral Efficiency Improvement: Energy and spectral efficiency can be improved by caching content at user devices, or by caching at base stations or edge routers to eliminate the backhaul bottleneck. Spectral efficiency and energy efficiency can benefit from wireless edge about 900 percent and 500 percent, respectively [4].

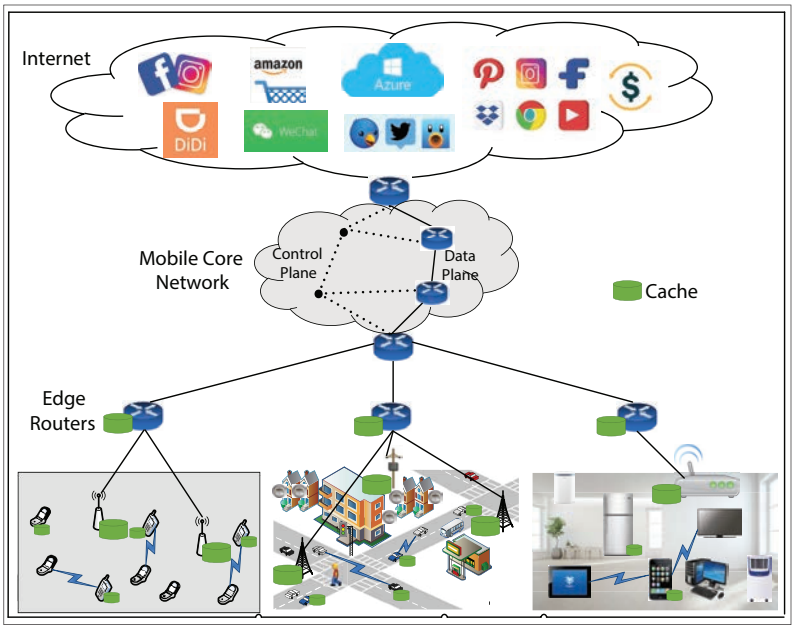

FIGURE 1. Mobile edge caching.

Where to Cache?

As depicted in Fig. 1, the caches can be placed on the devices at the network edge, including edge in wired network and edge in wireless network.

Caching at UE: Caching at UE is referred to as device-to-device caching. The devices are organized into clusters and directly communicate with each other using license-band or unlicense-band protocols. The requested content can be offered by some other UE within the same cluster that are caching the content.

Caching at Base Stations (BSs): Caching at BSs is referred to deploy caching at micro base stations (MBSs), small base stations (SBSs), pico base stations (PBS), and femto base stations (FBSs). Generally, MBSs have larger cache spaces and coverage areas than SBSs, and SBSs use cooperative caching to share content with each other. PBSs and FBSs are special SBSs. In the 5G era, C-RAN is an important architecture of 5G cellular networks [6]. C-RAN aggregates the computing capabilities to the baseband unit (BBU) pool, and deploys multiple radio remote heads (RRHs) to address network capacity and coverage issues. Cache spaces are deployed at both BBU and RRH that form a distributed edge caching architecture. At the wireless network edge, caching can be located at wireless routers for enabling various applications, for example, smart home and smart office.

In addition, the content can be cached at the wired network edge, that is, on edge routers.

What to Cache?

The aim of caching is to reduce backhaul traffic and service delay, so that the network providers prefer to cache all the content that may be requested by users. Nevertheless, the storage spaces of edge caches are limited and the scale of content is growing significantly [4]. Thus, it is important to determine what content should be cached by con-

With the development of ubiquitous networking, the connected devices are exposed to various cyber threats, ranging from generic threats, such as DoS attacks, wireless jamming, malware attacks, and man-in-the-middle attacks, to cache-specific threats, including cache poisoning attacks, cache pollution attacks, cache side-channel attacks, and cache deception attacks.

sidering popularity. In fact, only a small amount of content is extremely popular, while a long tail of content is accessed by a small portion of mobile users. "Cacheability" is changing over caching strategies and content types [5]. For example, it is unnecessary to cache the content that has been maintained on the neighboring caches, and some content with special types, for example, images and videos, may have the highest priority to be cached as they have the highest revisit rate. Moreover, only public and static content, such as images, CSS files, PDF documents, and videos, will be cached. The cached content is not user-specific.

How to Cache?

According to when to decide whether to cache content on edge nodes, caching can be classified as reactive caching and proactive caching. Reactive caching determines whether to cache a particular content after it has been requested, while proactive caching determines what content should be cached before it is requested. Both reactive caching and proactive caching make the decision based on the content types (e.g., files, software packages, and videos), content popularity (i.e., the ratio of the number of requests in a particular region), or the profiles of potential interested users (e.g., access patterns and mobility patterns). Content popularity and the profiles of interested users may vary with time. The cached content would be updated based on the caching policies or strategies. In general, the proactive caching and reactive caching include the following steps.

Content Placement/Update: The objectives of content placement and content update are consistent, that is, to determine the locations of content for maximizing cache-hit rate. An effective caching policy can accurately estimate the gains for caching content by evaluating current and future popularity and optimizing storage spaces and cache node selection. Therefore, in content placement and content update, one of the key issues is to estimate the popularity of content. User mobility, social associations, and preferences are essential criteria to predict content's popularity, and enable effective content placement and cache update to achieve high cache-hit rate.

Content Delivery: The UE requests the interested content from the service servers. After receiving content requests, edge nodes determine whether the requested content is cached or not by finding the unique identifier for every object, called cache key, in the cache. If a cache-hit happens, the edge node directly delivers the requested content to UE; otherwise, it fetches the content from the remote servers through the mobile core network.

Content Usage: The UE receives the requested content from the edge nodes or the remote servers. The UE needs to check that the content is current and it is from the correct service providers. The UE not only makes use of the obtained content to enjoy the corresponding services, but also serves as the cache node to offer content to other UE within the cluster.

Privacy, Security, and Trust

In this section, we introduce the potential cyber threats that corrupt security, privacy, and trust in mobile edge caching, and discuss the issues caused by these threats.

Threats in Mobile Edge Caching

With the development of ubiquitous networking, the connected devices are exposed to various cyber threats, ranging from generic threats, such as DoS attacks, wireless jamming, malware attacks, and man-in-the-middle attacks, to cache-specific threats, including cache poisoning attacks, cache pollution attacks, cache side-channel attacks, and cache deception attacks. The generic threats to edge networks are the same as the risks to the well known communication and information systems. For example, the DoS attacks flood edge nodes with superfluous requests in an attempt to make the services offered by the edge nodes unavailable to its intended UE. We omit the introduction of generic attacks and refer to [7] for the details. Cache-specific attacks are introduced as follows.

Cache Poisoning Attacks: An adversary inserts the forged or false content into the edge caches to deceive the users who request it. For example, by utilizing the vulnerability of cache keys, an adversary creates a cache key collision and manipulates the cache keys to cause a harmful or crafted response to the request from the targeted UE. The attacker may request a content with a query containing a malformed identifier, which may cause the remote server to crash. As a consequence, the false content is returned and cached on the edge node and served to other mobile users who request it. The cache poisoning attack could result in an attacker delivering false or error HTML pages, javascript files, stylesheets, images, and videos to the mobile users instead of legitimate content. Therefore, the cache poisoning attack threatens the availability and the correctness of network resources. The users may be successively deceived until the poisoned cache is updated. The success of this attack relies on the exploitable vulnerabilities on the edge caches.

Cache Pollution Attacks: An attacker pollutes caching balance by sending large amounts of fabricated content to the cache and thereby tricking the cache into caching non-popular content. Under this attack, the cache-hit rate of the legitimate users is degraded, and their response time is increased. There are two types of cache pollution attacks, that is, location-disruption attacks and false-locality attacks. In the location-disruption attack, an adversary frequently requests some non-popular content and squeezes out the popular content in caches. In the false-locality attack, an adversary continually requests a set of non-popular content and inserts it into the cache.

Cache Side-Channel Attacks: By observing and measuring the activities of edge caches, such as response time, power consumption, cache access, and returned fault, an adversary can learn private information about mobile users and cached content. Cache side-channel attacks can be divided into different types based on the knowledge that the adversaries have. For example, in cache-timing attacks, by comparing the response time, an adversary can determine whether particular content has been cached. In content deduplication attacks, a copy-on-write content fault reveals the fact that the requested content is deduplicated and users must have requested the identical content. In cache privacy attacks, an adversary can learn the access history and other possible private information of a particular user, if they share the same edge cache.

Cache Deception Attacks: The attacker first tricks the target user into making a request to the private content that it is not permitted to access. The attacker then submits the same request, and edge caches can return the requested content containing the previously private data. For example, an attacker crafts a URL request that points to the bank account information of a victim, but appends to it a non-exist path of a static image, and tricks the victim into making this request. The server will ignore the invalid suffix and return the account details, and the cache node will maintain the content which is identified as a static image. The attacker can make the same request to retrieve the account details. Thus, the private data of the victim will be disclosed.

Cache poisoning attacks and cache pollution attacks may corrupt the integrity of cached content, cause the unavailability of requested data, and degrade the trust of mobile users on the edge caching services. Cache side-channel attacks and cache deception attacks compromise the confidentiality of content on caches and invade the privacy of mobile users. These threats on security, privacy, and trust may compromise all the appealing benefits of mobile edge caching. Security, privacy, and trust are different requirements for the reliability of mobile edge caching, but they are all interrelated. If security or privacy are not protected in content placement or content delivery, the trust in data usage cannot be built. Therefore, it is essential to preserve security, privacy, and trust in the entire procedure of mobile edge caching services.

Security, Privacy, and Trust Challenges

We examine the typical issues in the whole procedure, from content placement to content delivery and content usage.

Privacy in Content Placement: Compared to the increasing data volumes in mobile networks, the storage space on edge nodes is always limited. It is impossible to cache all data locally. Hence, identifying the optimal set of contents that maximizes the utility of each cache becomes crucial. A common approach of content set identification for placing content is based on content popularity. To estimate popularity, a machine-learning model [8, 9] is needed to analyze historical access records, social association, and mobility patterns of mobile users. However, machine learning is a double-edged sword. While enabling the discovery of knowledge from a large volume of data, it may invade user privacy by exposing potentially sensitive information. Such privacy invasion may trigger serious results. For example, the exposure of mobile users' preferences may bring various unwelcome advertisements and promotions. Therefore, questions that need to be addressed include how to achieve privacy-preserving content placement based on content popularity and user preferences? How to predict content popularity using machine learning without exposing the privacy of mobile users? It is essential to explore privacy-preserving techniques and design advanced solutions as a long-term vision for content placement in mobile edge caching.

Security in Content Delivery: UE searches and retrieves content from the correct caches if they are on edge nodes; otherwise, it has to request the data from remote servers. Knowing what content is stored on the edge nodes at any given moment is essential, while maintaining this knowledge is challenging in a distributed cache network. The Bloom filter and its variants are common approaches for space-efficient approximate membership representation, a useful technique to indicate whether an element is a member of a large group. A shortcoming of such approaches is the false positive rate, that is, a given item is indicated to be cached but actually not there. Othello hashing is used to design a memory-efficient and fast localization method of edge nodes [10] to reduce the rate of false locating, but Othello construction and update operations are expensive. Besides, end-to-end encryption is the major method for securing data transmission on public channels (e.g., the transport layer security (TLS) protocol), while protecting content maintained on edge nodes against the cache side-channel attacks and the cache deception attacks, but it seemingly signifies the death of edge caching. The content cached on edge nodes is encrypted for the first requesting mobile user, such that the subsequent UE cannot decrypt the fetched data from the edge nodes. CryptoCache [11], a secure content transmission protocol, was designed based on symmetric encryption to enable caching of encrypted content on an untrusted edge node, but the cloud server is still needed even if the requested content is cached. Content partitioning and scrambling [12] is an efficient solution to achieve content confidentiality but cannot provide sufficient security guarantees, that is, semantic security; attribute-based encryption [13] offers high data security, but brings heavy computational overhead. Therefore, questions include: How to design effective content locators on edge nodes for optimal content exploration? How to design efficient data encryption schemes for secure content delivery without involving remote cloud servers? Exploring new models, techniques, and approaches to guarantee effective content exploration and reason about the data confidentiality during content delivery against the cache side-channel attacks and the cache deception attacks is deserved to focus on.

Trust in Content Usage: Mobile users recover their content of interest after receiving the encrypted data from edge caches. Due to the security threats, mobile users may have huge concerns regarding the trustworthiness of their obtained content, for example, whether they have been corrupted by ill-intentioned edge nodes, poisoned by adversaries with cache poisoning attacks, or are unavailable when requested due to the cache pollution attacks. The trust decline due to the violation of authenticity, integrity, and availability is one of the major obstacles to the widespread use of mobile edge caching. The sophisticated techniques (e.g., content checking, message authentication, and digital signature) may not be practically feasible to address these issues as a whole. For example, the digital signature can guarantee the authenticity and integrity of the content, but it cannot effectively prevent or detect cache poisoning attacks, since in the cache poisoning attack, the response is from the correct server, but the content is not the one requested. Moreover, they bring extra computational and communication overheads to both edge nodes and mobile users. These additional burdens may eventually ruin the major advantage of edge caching, that is, low-latency service response. Without efficient integrity verification,

Security, privacy, and trust are different requirements for the reliability of mobile edge caching, but they are all interrelated. If security or privacy are not protected in content placement or content delivery, the trust in data usage cannot be built. Therefore, it is essential to preserve security, privacy, and trust in the entire procedure of mobile edge caching services.

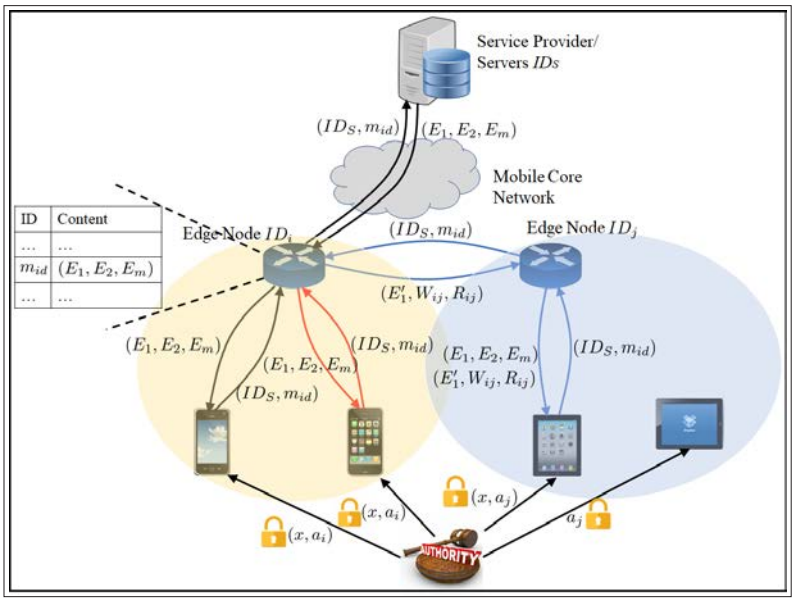

FIGURE 2. Model of SOLEK.

the trustworthiness of cached content depends on the honesty of edge nodes, but trust evaluation in a distributed edge network is not straightforward. Blockchain can potentially be leveraged to achieve distributed trust management on behalf of a decentralized, transparent and public ledger, but this research area remains a series of inherent issues, for example, privacy leakage and storage overload. Potential questions include, how to design efficient content verification schemes to detect corrupted content, and how to build a trustful edge-caching framework based on blockchain and smart contracts. These problems, if not properly resolved, will impede the success of mobile edge caching.

SOLEK

We propose SOLEK, a service-oriented and location-based efficient key distribution protocol based on certificateless proxy re-encryption [14] for secure content delivery in mobile edge caching. SOLEK enables a mobile user to decrypt the ciphertexts of the content cached on the edge node. The decryption capability is delegated based on the service type ID_S and the position of the edge node ID_i, such as GPS data. Specifically, a mobile user is distributed with two separate secret-public key pairs. One is related to the service type ID_S that is given in the service registration, and the other is about the edge node ID_i that is allocated when entering the coverage area of ID_i. Also, the mobile user can access the content cached by a neighboring edge node ID_j of ID_i by utilizing the proxy re-encryption technique. The model of SOLEK is shown in Fig. 2, and the detailed algorithms in SOLEK are described in Fig. 3.

In SOLEK, a trusted authority (TA) first executes the Setup algorithm to bootstrap the whole protocol. The TA can be the key generation center (KGC) in the identity-based cryptosystem. With the input of the security parameter κ, the **Setup** algorithm outputs the public parameter $param = (G, p, P, P_{pub}, H_1, H_2, H_3, H_4, H_5)$ and the master secret key s. Then, with the inputs $(param, s, ID_S)$, TA runs the **ServKeyExt** algorithm to generate the service secret key y and the service public key $SPK = (X, Y, R)$. SPK is public and y is sent to the mobile users who register the service ID_S through secure channels. When receiving y, the mobile users can verify whether y is correct by checking $RP \stackrel{?}{=} Y + H_1(X, Y, ID_S)P_{pub}$. Also, with the inputs $(param, s, ID_i)$, TA executes the **EdgeKeyExt** algorithm to generate the edge secret key a_i and the edge public key $EPK = (A_i, B_i, D_i)$ for the edge node ID_i. EPK is public and a_i is sent to the mobile user who enters the coverage area of ID_i through a secure channel. When receiving a_i, the mobile user can verify whether a_i is correct by checking $D_iP \stackrel{?}{=} B_i + H_1(A_i, B_i, ID_i, t_i)P_{pub}$. Here, t_i is the current time slot, and the edge secret-public key pair can be updated by TA periodically. If a mobile user U requests the data m of the service ID_S in the coverage area of ID_i, U sends ID_S and the file name m_{id} to ID_i. ID_i checks whether m is cached or not. If not, that is, cache miss, the request is forwarded to the server. The server chooses a session key k to encrypt m to generate E_m using a block cipher, for example, AES, and runs the **Encrypt** algorithm to encrypt k to obtain (E_1, E_2). (E_1, E_2, E_m) is returned to U and also cached on ID_i. U runs the **Decrypt** algorithm to decrypt (E_1, E_2) and obtain k for data decryption. If yes, it means cache hit, that is, (E_1, E_2, E_m) is cached on ID_i, U can directly fetch (E_1, E_2, E_m) from ID_i. Subsequently, if m is requested by anther mobile user U' in the coverage area of ID_j, a neighboring edge node of ID_i, ID_i runs the **ReKeyGen** algorithm to generate the re-encryption key $RK = (\alpha_{ij}, W_{ij}, R_{ij})$, and the **ReEncrypt** algorithm to transform (E_1, E_2, E_m) to $(E_1, E'^1, E_2, W_{ij}, R_{ij}, E_m)$ for U'. Finally, U' fetches $(E_1, E'^1, E_2, W_{ij}, R_{ij}, E_m)$ from ID_j, and runs the **ReDecrypt** algorithm to decrypt (E_1, E'_1, E_2) and obtain k for data decryption.

In SOLEK, two properties are provided, that is, content confidentiality and content sharing. To ensure the content confidentiality in SOLEK, two types of attackers should be considered. The one has registered the service ID_S but is not in the coverage area of ID_i; the other is in the coverage area but does not register the service ID_S. If neither can compromise SOLEK, other attackers, for example, non-registered ID_S and outside ID_i attackers, cannot break SOLEK. SOLEK has two levels of ciphertexts, (E_1, E_2, E_m) and $(E_1, E'_1, E_2, W_{ij}, R_{ij}, E_m)$. We ensure the confidentiality of cached content, that is, each type of attacker cannot gain any knowledge from any level of ciphertexts. In order to formalize plaintext leakage from ciphertexts, the distinguishability has been defined that an attacker cannot identify the plaintext choice given an encryption of a plaintext randomly chosen from a two-element plaintext space determined by the attacker with probability significantly better than that of random guessing. According to this property, the security of SOLEK can be reduced to the hard assumption, that is, the Decisional Diffie-Hellman (DDH) assumption.

SOLEK achieves efficient content sharing among mobile users in the same area for the service of common interest. It provides higher security guarantees than the scheme in [12], under the assumption that all the registered mobile users will not proactively expose their interested content. Also, SOLEK avoids interactions with the server to retrieve the session key if cache hit compared with CryptoCache [11]. Attribute-based encryption (ABE) provides tremendous potential for fine-grained data sharing based on multiple attributes, while SOLEK surpasses the ABE-based data sharing schemes, such as PCWX [13] and PWXL [15], in terms of computational effi-

Setup: Given a security parameter κ, the algorithm generates the group G of a prime order p. Let P be a generator of G, It picks $s \in Z_p^*$ to compute $P_{pub} = sP$ and chooses cryptographic hash functions $H_1 : G^2 \times \{0,1\}^{l_{ID}} \to Z_p^*$, $H_2 : Z_p^* \times Z_p^* \to Z_p^*, H_3 : \{0,1\}^{l_{ID}} \times \{0,1\}^{l_{ID}} \times G \to Z_p^*$, $H_4 : \{0,1\}^{l_1} \times \{0,1\}^{l_2} \to Z_p^*$, and $H_5 : G \times G \to \{0,1\}^{l_1+l_2}$.

ServKeyExt: Given a service identtity ID_S, the algorithm randomly selects $x, y \in Z_p^*$ and computes $X = xP, Y = yP$, $Z = H_1(X, Y, ID_S)$, and $R = y + sZ \mod p$.

EdgeKeyExt: Given an edge identity ID_i, the algorithm randomly selects $a_i, b_i \in Z_p^*$ and computes $A_i = a_iP$, $B_i = b_iP$, $C_i = H_1(A_i, B_i, ID_i, t_i)$, and $D_i = b_i + sC_i \mod p$.

ReKeyGen: The algorithm randomly picks $\alpha_{ij} \in Z_p^*$ and computes $\beta_{ij} = a_i\alpha_{ij}^{-1} \mod p$, $w_{ij} = H_2(\alpha_{ij}, \beta_{ij})$, $W_{ij} = w_{ij}A_j$, and $R_{ij} = H_3(ID_i, ID_j, w_{ij}P) \oplus \beta_{ij}$.

Encrypt: The algorithm randomly selects $\sigma \in \{0,1\}^{l_2}$ and computes $r = H_4(k, \sigma)$, $E_1 = rP$, and $E_2 = H_5(rA_i, rX) \oplus (k||\sigma)$.

ReEncrypt: The algorithm computes $E_1' = \alpha_{ij}E_1$.

Decrypt: The algorithm computes $(k||\sigma) = E_2 \oplus H_5(a_iE_1, xE_1)$ and verifies $E_1 \stackrel{?}{=} H_4(k, \sigma)P$.

ReDecrypt: The algorithm computes $\beta_{ij} = H_3(ID_i, ID_j, W_{ij}a_j^{-1}) \oplus R_{ij}$, $(k||\sigma) = E_2 \oplus H_5(\beta_{ij}E_1', xE_1)$, and verifies $E_1 \stackrel{?}{=} H_4(k, \sigma)P$.

FIGURE 3. SOLEK algorithms.

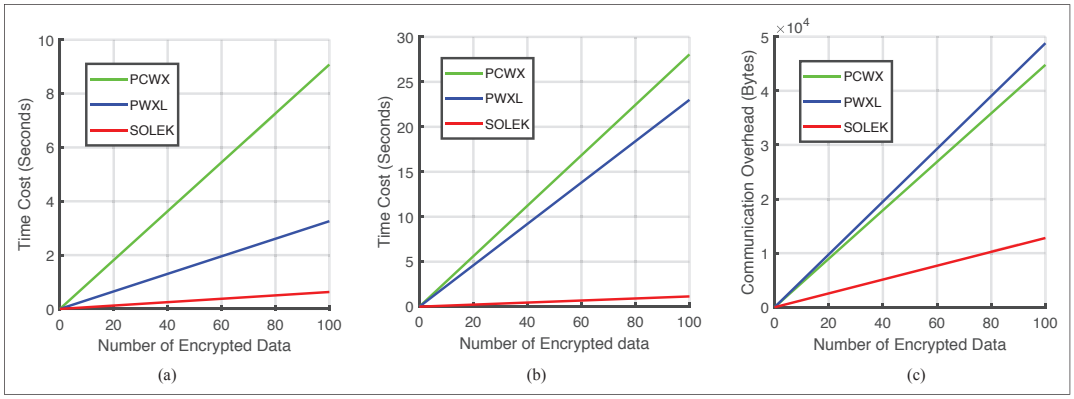

FIGURE 4. Performance comparison: a) time cost for server on encryption; b) time cost for UE on decryption; c) communication overhead.

ciency. The time costs of content encryption and decryption in PCWX [13], PWXL[15], and SOLEK are demonstrated in Fig. 4. Since the cached content is shared based on the service type and edge location, two attributes are considered in PCWX [13] and PWXL [15]. The experiments are conducted on a smartphone of HUAWEI MT2-L01 with Kirin 910 CPU and 1.25MB memory. The Barreto-Naehrig curve is utilized and the security parameter k=128. Figures 4a and 4b demonstrate that SOLEK is much more efficient than PCWX [13] and PWXL [15] on data encryption and decryption; Fig. 4c illustrates the communication overhead of SOLEK in content transmission is less than that of PCWX [13] and PWXL [15].

Future Directions

In this section, we further explore the potential techniques to address the security, privacy, and trust challenges in mobile edge caching, which are expected to attract attention in future research.

Privacy-Preserving Content Placement

Federated learning [9] can be utilized to predict content popularity. In federated learning, the model parameters are locally trained on edge nodes, and periodically exchanged and updated through unreliable network channels between the edge nodes and the remote server. The private information of mobile users does not need to be sent to the server. However, the information is still vulnerable to being disclosed to edge nodes during historical data collection. Central differential privacy (DP), multi-party computation, and homomorphic encryption are possible to protect the confidentiality of content, but each has inherent weaknesses. The central DP relies on a trusted "aggregator" that holds the sensitive data of all individuals for protecting their privacy, multi-party computation requires extra interactions between local edge nodes, and homomorphic encryption causes heavy computational overhead for UE. Privacy leakage might possibly be prevented by utilizing local differential privacy (LDP), in which individuals randomly perturb their inputs to offer plausible deniability of their data without a trusted party. However, this technique is still in its infancy, although a few of approaches have been proposed to achieve strong privacy preservation in federated learning that support shallow models such as logistic regression. These approaches suffer from the intrinsic limitations that they either cause poor estimation accuracy or only focus on simple datasets or tasks. Moreover, they incur a significant privacy issue of the local model, as they expose all internal model states to white-box inference attacks. The major challenge of LDP is that it introduces large noise to local training in federated learning, such that the cache-hit rate is quite low. Therefore, it is essential to design new data pre-processing mechanisms (e.g., data encoding and matrix permutation) or integrate distribution estimation techniques for privacy-preserving federated learning in content placement.

Efficient and Secure Content Delivery

The Cuckoo filter is an efficient data structure for membership check to locate the requested content on edge nodes for mobile users. Compared

Without efficient integrity verification, the trustworthiness of cached content depends on the honesty of edge nodes, but trust evaluation in a distributed edge network is not straightforward. Blockchain can potentially be leveraged to achieve distributed trust management on behalf of a decentralized, transparent and public ledger, but this research area remains a series of inherent issues.

To reduce storage costs, the approaches to compressing the elements at the same indices of the Cuckoo filters and encapsulating the content indices for the edge node deserve investigation. In addition, the efficient connection checking mechanisms are required to enable an edge node to check connections with its neighboring edge nodes periodically and update TTL values.

with the Bloom filter, the Cuckoo filter supports the removal of unpopular content and achieves constant lookup cost for edge nodes. However, false positive matches are still possible. Thus, the naming policies of content, including flat naming and hierarchical naming, should be extended to avoid false positives. Furthermore, in distributed edge caching, it is challenging to locate the content in the neighboring edge nodes and determine the time-to-live (TTL) value of a content request. To ensure efficient content localization, an edge node should maintain all Cuckoo filters of the edge nodes within the TTL range that consume the linear storage space of the edge node with the number of reachable edge nodes. To reduce storage costs, the approaches to compressing the elements at the same indices of the Cuckoo filters and encapsulating the content indices for the edge node deserve investigation. In addition, the efficient connection checking mechanisms are required to enable an edge node to check connections with its neighboring edge nodes periodically and update TTL values.

Trustful Content Usage

The content is cached on an edge node once a mobile user requests, the corresponding signature of the server can be maintained on the public blockchain for authentication and integrity verification. Subsequently, mobile users can check the content integrity based on the signature on the blockchain when needed. Thus, it is unnecessary to forward the signature to the mobile users. Also, secure binding of cache keys and content (i.e., key-data integrity) should be achieved against potential replace attacks and poisoning attacks. The zero-knowledge proof can be used to ensure that the content and the cache keys on the blockchain are identical to those cached on the edge nodes. This proof of consistency is essential to ensure content reliability and integrity. Moreover, the behaviors of fog nodes should be regulated. Smart contracts are designed to enable edge nodes to make commitments and enforce punishment on untrusted or rogue edge nodes in case any misbehavior is detected. However, the usage of smart contracts can only guarantee enforcement, not privacy. That is, the privacy of mobile users will be disclosed. Although the variants of zero-knowledge proofs, such as Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARKs) and Zero-Knowledge Succinct Transparent Argument of Knowledge (zk-STARKs) can offer privacy-preserving payments and interactions, the open problems, including the efficiency and the trusted setup in zk-SNARKs and proof size reduction in zk-STARKs, are receiving a lot of attentions in research.

Conclusion

In this article, we have studied the security, privacy, and trust issues in mobile edge caching. Specifically, we have answered four key questions in mobile edge caching to have a better understanding, and introduced its security threats to the content in caches. The challenges of privacy, security, and trust in content placement, content delivery, and content usage of network edge caching are presented, respectively. In response to efficient and secure content delivery, a service-oriented and location-based efficient key distribution protocol (SOLEK) is proposed based on certificateless proxy re-encryption. Finally, several interesting research issues and potential techniques have been identified to shed light on further research about secure, privacy-preserving, and trustful mobile edge caching.

References

[1] C. Stergiou *et al.*, "Secure Integration of IoT and Cloud Computing," *Future Generation Computer Systems*, vol. 78, no. 3, 2018, pp. 964–75.

[2] Y. Hu *et al.*, "Mobile Edge Computing — A Key Technology Towards 5G," ETSI White Paper, vol. 11, no. 11, 2015, pp. 1–16.

[3] E. Zeydan *et al.*, "Big Data Caching for Networking: Moving from Cloud to Edge," *IEEE Commun. Mag.*, vol. 54, no. 9, 2016, pp. 36–42.

[4] D. Liu *et al.*, "Caching at the Wireless Edge: Design Aspects, Challenges, and Future Directions," *IEEE Commun. Mag.*, vol. 54, no. 9, 2016, pp. 22–28.

[5] X. Wang *et al.*, "Cache in the Air: Exploiting Content Caching and Delivery Techniques for 5G Systems," *IEEE Commun. Mag.*, vol. 52, no. 2, 2014, pp. 131–39.

[6] P. Yang *et al.*, "Catalyzing Cloud-Fog Interoperation in 5G Wireless Networks: An SDN Approach," *IEEE Network*, vol. 31, no. 5, 2017, pp. 14–20.

[7] R. Lu *et al.*, "5G Vehicle-to-Everything (V2X) Services: Gearing Up for Security and Privacy," *Proc. IEEE*, vol. 108, no. 2, 2020, pp. 373–89.

[8] L. Xiao *et al.*, "Security in Mobile Edge Caching with Reinforcement Learning," *IEEE Wireless Commun.*, vol. 25, no. 3, 2018, pp. 116–22.

[9] X. Wang *et al.*, "In-Edge AI: Intelligentizing Mobile Edge Computing, Caching and Communication by Federated Learning," *IEEE Network*, vol. 33, no. 5, 2019, pp. 156–65.

[10] I. Cohen *et al.*, "Access Strategies for Network Caching," *Proc. IEEE INFOCOM*, 2019, pp. 1–9.

[11] J. Leguay *et al.*, "CryptoCache: Network Caching with Confidentiality," *Proc. IEEE ICC*, 2017, pp. 1–6.

[12] Z. Su *et al.*, "A Secure Content Caching Scheme for Disaster Backup in Fog Computing Enabled Mobile Social Networks," *IEEE Trans. Industrial Informatics*, vol. 14, no. 10, 2018, pp. 4579–89.

[13] J. Pan *et al.*, "Secure Data Sharing Scheme for VANETs Based on Edge Computing," *EURASIP J. Wireless Commun. Networking*, article no. 169, 2019.

[14] S. Selvi *et al.*, "An Efficient Certificateless Proxy Re-Encryption Scheme without Pairing," *Proc. ProvSec*, 2017, pp. 413–33.

[15] Y. Pu *et al.*, "An Efficient and Recoverable Data Sharing Mechanism for Edge Storage," *Proc. WASA*, 2019, pp. 247–59.

Biographies

Jianbing Ni [M'18] is currently an assistant professor with the Department of Electrical and Computer Engineering, Queen's University, Kingston, Canada. He received his Ph.D. degree in electrical and computer engineering from the University of Waterloo, Canada, in 2018. His research interests are applied cryptography and network security, with a current focus on edge computing, mobile crowdsensing, Internet of Things, and blockchain technology.

Kuan Zhang [S'13, M'17] received the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Canada, in 2016. He was also a postdoctoral fellow with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. Since 2017, he has been an assistant professor in the Department of Electrical and Computer Engineering, University of Nebraska-Lincoln, USA. His research interests include security and privacy for mobile social networks, e-healthcare systems, cloud computing and cyber physical systems.

Athanasios V. Vasilakos is with the School of Electrical and Data Engineering, University of Technology Sydney, Australia, and with the Department of Computer Science, Electrical and Space Engineering, Lulea University of Technology, Lulea, 97187, Sweden. He served or is serving as an editor for many technical journals, such as the *IEEE Transactions on Network and Service Management*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Cybernetics*, *IEEE Transactions on NanoBioscience*, *IEEE Transactions on Information Technology in Biomedicine*, *ACM Transactions on Autonomous and Adaptive Systems*, and *IEEE JSAC*. He is a Web of Science 2017, 2018, 2019, and 2020 Highly Cited Researcher.