Poster: An IoT Data Communication Framework for Authenticity and Integrity

Xin Li*, Huazhe Wang*, Ye Yu†, Chen Qian*
Email: {xli178, hwang137, cqian12}@ucsc.edu , ye.yu@uky.edu
*Department of Computer Engineering, University of California Santa Cruz
†Department of Computer Science, University of Kentucky

I. Introduction

Internet of Things has been widely applied in everyday life, ranging from transportation, healthcare, to smart homes. As most IoT devices carry constrained resource and limited storage capacity, sensing data need to be transmitted to and stored at resource-rich platforms, such as a cloud. IoT applications retrieve sensing data from the cloud for analysis and decision-making purposes. Ensuring the authenticity and integrity of the sensing data stored in third-party clouds is essential for the correctness and safety of IoT applications.

In this paper we present the design of an IoT data communication framework involving the three key entities: *sensing devices*, *cloud* and *data applications*. We assume only sensing devices and data applications are trustworthy and any entities in between are subject to attack. Authenticity and integrity should be *verifiable* by data applications. Moreover, different applications may have different requirements on sending data granularity. Even if the cloud can store up to 100 records, some applications only retrieve part of them, e.g., 10 records, due to bandwidth/memory limit or application requirements. In addition, these 10 records should have verifiable authenticity, integrity, and uniformity. We call this feature as *partial data retrieval*. We do not address the issue of data confidentiality or privacy in this poster.

II. System Design

A. Existing Signature Schemes

Digital signature is a widely used method to protect data authenticity and integrity. However, applying digital signature to every sensing record, called the *Sign-each* method, is not practical, because public-key encryption/decryption are considered slow and expensive, especially for sensing devices with limited resource. A more efficient method, *concatenate*, is to compute the message digest for a large number of records and sign once. This approach requires each sensing device to cache all records and has the all-or-nothing feature: if some applications only require part of the records, the signature cannot be verified. A well-known method to sign a data stream is hash chaining [5]. It does not fit the IoT framework either, because partial data retrieval will break the chain and make the signature unverifiable.

To address the aforementioned problems when applying digital signature in the IoT scenario, we propose two signature schemes. 1) the Dynamic Tree Chaining (DTC) that is developed based on Merkle tree [6] [7]. DTC serves as the baseline in this poster. 2) a novel signature scheme specifically designed for the IoT data communication framework, called Geometric Star Chaining (GSC). GSC outperforms DTC in terms of throughput and memory consumption. GSC provides verifiable uniformity while DTC does not. The comparison of different signature schemes is listed in TABLE. I.

TABLE I. Overall comparison of different signature schemes.

Signature Scheme	Computation efficiency	Partial data retrieval	Constant space cost	Sampling uniformity
Sign-each	X	✓	✓	X
Concatenate	✓	X	X	X
Hash chaining	✓	X	✓	X
DTC	✓	✓	X	X
GSC	✓	✓	✓	✓

B. Dynamic Tree Chaining (DTC)

We start from the Tree chaining designed by Wong and Lam [7], one variation of Merkle tree [6]. The digest of each event report is one leaf node in binary *authentication tree*. The value of the internal node is computed as the hashing of the concatenation of its two children. As a result, the root summarizes all the leaf nodes. The root node is regarded as the block digest and is signed by the private key to create the block signature. The verification process is on a per-event basis. In order to verify the integrity/authenticity of an event e, the verifier requires the block signature and the sibling nodes in the path to the root, which are all appended to event e.

The expensive asymmetric encryption operation is amortized to all events in one authentication tree and thus tree chaining is computational efficient. More importantly, since every single event is verifiable, it is fully compatible with partial data retrieval without resource waste. The most severe issue that impedes the adoption of the original tree chaining in IoT environment is that all events should be buffered in the sensing device before the authentication tree is built, since each event ought to be appended auxiliary authentication information from the authentication tree.

We observe that introducing the cloud can greatly reduce the memory footprint at sensing devices. The sensing device only maintains the message digest of each event and sends all events to the cloud directly without caching. The authentication tree grows in an online fashion and the sensing device transmits to the cloud internal nodes that are no longer needed for calculating the remaining authentication tree. The space complexity in the sensing device to host nodes of authentication tree is reduced from $O(n)$ to $O(\log n)$, where n denotes the number of events in one epoch.

Once the buffer in the sensing device is full, the root node in the authentication tree is signed and the remaining nodes are all flushed to the cloud to spare space for upcoming events. Therefore, one sensing device may apply digital signature more than once in one single epoch. As a result, the buffer space constrains the performance of DTC, which is a particularly severe problem in IoT environment where most devices possess little buffer space. More importantly, *DTC provides no verifiable uniformity for event sampling and partial data retrieval*, because data selection is completely executed in the cloud. If the cloud selects event samples or the partial data with bias, data applications are unaware of it.

C. Geometric Star Chaining (GSC)

We present a more efficient and secure data communication framework in this poster, called Geometric Star Chaining (GSC). The basic idea of GSC is inspired by the observation that any arbitrary fraction value can be represented or closely approximated by a few number of binary digits. For instance, $5/8 = (0.101)_2$. Thus, partial data with sample rate p, where $p = \sum 2^{-b_i}$ is equivalent to the union of multiple data blocks each corresponds to one set bit in the binary representation. One data block is called a *sample block* in this poster. For instance, to retrieve a sampled data from all sensing data from a device within an epoch with sampling rate $5/8$, the cloud can send the data application two blocks containing (approximately) $1/2$ and $1/8$ of the events respectively.

The number of events included in the sample blocks follows *geometric distribution*. Each sample block should draw events uniformly from the IoT data stream. In order to ease the presentation of how sample blocks form, we define a set of successive numerical intervals $\{S_i\}$ where $S_i \triangleq \{x \in \mathbb{R} : 2^{-i-1} < x \leq 2^{-i}, i \in \mathbb{N}\}$. On receiving a new event e, the sensing device computes which numeric interval in $\{S_i\}$ that $h(e)$ falls in and event e is inserted into the corresponding sample block, where $h(\cdot)$ is a non-cryptographic uniform hashing function and $\forall x : 0 \leq h(x) \leq 1$.

Events within a same data block are either completely retrieved or not retrieved at all. Thus we can view each data block as an atomic "giant event". GSC computes one message digest for every block and concatenates these digests to a single digest for digital signature. The digest of each sample block is computed in an online fashion. One variable D_i is allocated to each sample block to capture the newest value of the message digest. Suppose a new event e is observed at the device which belongs to the ith sample block. The message digest D_i is updated as $D_i = h(h(e)||D_i)$. This online updating proceeds until the end of the epoch. In the end, the concatenate approach is applied to all the message digests $\{D_i\}$. Since the output of any hashing function is one finite-length bit sequence, the number of sample blocks is bounded and thus space cost for this signature scheme at the sensing device is constant.

III. Implementation and Evaluation

We implement GSC as well as another two alternative signature schemes, sign-each approach and DTC for comparison. We utilize OpenSSL [3] to implement the widely used asymmetric encryption algorithm DSA [2]. MD5 is leveraged as the message digest function. Particularly for GSC, the non-cryptographic 64-bit hash function xxHash [4] classifies events into sample blocks. The prototype emulation is conducted on a quadcore@3.40G Linux desktop with 32GB memory but only one core is used.

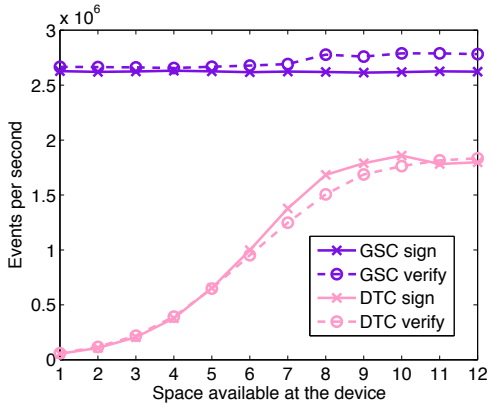

Fig. 1. Signing/verifying throughput comparison.

We evaluate the performance by feeding the signer/verifier a set of public IoT data traces [1]. The 7 data sources each divided into 90 epochs are the input to the singing phase of the signature scheme, whose output feeds the verifying phase afterwards. Only one encryption operation is required in a single epoch for each data source. The space cost at both the signer and the verifier to host the events themselves is orthogonal to the choice of signature scheme. In order to simplify the presentation, we refer one unit of space cost as the memory space used for storing one message digest.

We measure the performance of two signature schemes with given space limit in the signer. To focus on the impact of the space issues at the signer side, we allocate enough free space to the verification process. For DTC, once the buffer in the signer is full, the root node in the authentication tree is signed and the remaining nodes are flushed to the cloud to make room for upcoming events. Thus, lacking space in the signer may lead to multiple expensive encryption operations in one epoch. Furthermore, the same number of decryption operations are also needed at the verifier side. Fig. 1 illustrates the signing/verifying performance comparison between GSC and DTC under varied space available. Both signing and verifying performance of DTC are capped by available memory at the signer, whereas GSC runs at full speed all the time. The performance of sign-each approach is not shown in Fig. 1 because it is more than 50X slower than the other two.

References

[1] http://traces.cs.umass.edu/index.php/Smart/Smart.

[2] Dsa. http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf.

[3] Openssl. https://www.openssl.org/.

[4] xxhash. http://www.xxhash.com/.

[5] R. Gennaro and P. Rohatgi. How to sign digital streams. In *Crypto*, 1997.

[6] R. C. Merkle. A digital signature based on a conventional encryption function. In *Proc. of CRYPTO*, 1987.

[7] C. K. Wong and S. S. Lam. Digital signatures for flows and multicasts. In *Proc. of IEEE ICNP*, 1998.