Design-flow Methodology for Secure Group Anonymous Authentication

Rashmi Agrawal, Lake Bu[†], Eliakin Del Rosario, Michel A. Kinsy
Adaptive and Secure Computing Systems (ASCS) Laboratory, ECE Department, Boston University
{rashmi23, edelrosa, mkinsy}@bu.edu
[†]The Charles Stark Draper Laboratory, Inc.
[†]{lbu}@draper.com

***Abstract*—In heterogeneous distributed systems, computing devices and software components often come from different providers and have different security, trust, and privacy levels. In many of these systems, the need frequently arises to (i) control the access to services and resources granted to individual devices or components in a context-aware manner and (ii) establish and enforce data sharing policies that preserve the privacy of the critical information on end users. In essence, the need is to authenticate and anonymize an entity or device simultaneously, two seemingly contradictory goals. The design challenge is further complicated by potential security problems, such as man-in-the-middle attacks, hijacked devices, and counterfeits. In this work, we present a system design flow for a trustworthy group anonymous authentication protocol (GAAP), which not only fulfills the desired functionality for authentication and privacy, but also provides strong security guarantees.**

***Index Terms*—authentication, anonymity, group, counterfeit-resistant, double blindness**

I. Introduction

With the emergence of Internet of Things (IoT) applications and cloud computing, a new class of security and privacy concerns has arisen. In current Systems-on-Chip (SoCs) and distributed computing systems, the devices, software components, processing elements, and intellectual property (IP) cores may have different provenances, security, trust, and privacy protection levels. In these heterogeneous computing systems, authentication at the group granularity can provide anonymity. When devices are partitioned by groups, it is possible for a device to authenticate itself as a genuine member of a group with certain privileges, without disclosing information about itself, as long as individual identities cannot be inferred from the group identity or tag. Therefore, the monitoring, tracking, recording, and analysis of activities or behaviors – *i.e.*, an access to a particular compute resource in the system – of specific devices or entities are rendered impossible. In short, the group members demand certain privileges without an obligation to disclose their full identities. Such a scenario is illustrated in Fig. 1.

The concept of a "group" can stand for a home with IoT devices, a department or team at a company with multiple employees, or an army unit, etc. For example, in a company of a certain size, different room or building access policies may apply to different employees and departments. While the company needs to configure its access cards to enforce these access policies, this should be done in a manner that provides employees a certain level of privacy. Similarly, an IoT device in a smart home may need a key piece of data to perform its function. For the request to be validated, the device should provide, to a server or another IoT device, its credential(s) or attribute(s) to a smart home. Ideally, it should not have to reveal other auxiliary information, such as its identity key, model, or usage, in order to preserve the privacy of its owner. In the same vein, for bare metal cloud-based computing, a service provider may avoid disclosing the single unique hardware identity to all the users ever assigned to that machine. A derived, user-specific, one-time access key or certificate may not be sufficient, since the user also needs to attest that his/her application is running directly on the bare metal matching his/her payment level, and not in a machine with lower specifications or even a virtual machine.

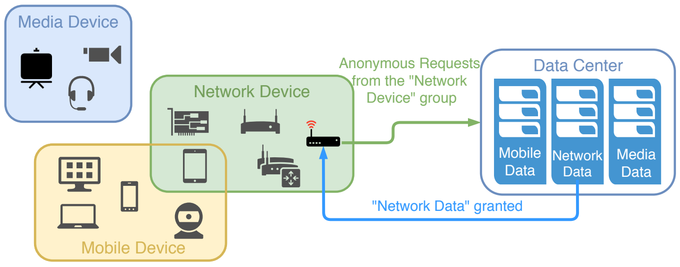


Fig. 1: When a request is made from a device in a certain group, say, the "Network Device" group, the data center is able to verify the group attribute of the request. So that the "Network Data" which can only be accessed by devices in that group is granted. Meanwhile, for privacy reasons, the device's individual identity is kept from disclosure during the entire process.

In terms of security challenges, there is a fundamental problem with using a single identifier – even an unclonable one – to authenticate a device directly. The identifier is static and thus, subject to replay attacks when there is a man-in-the-middle adversary. Similarly, using a single identifier to derive multiple certificates exposes the authentication protocol to certificate impersonation attacks [1]. In addition, the current single identifier approach does not directly support authorization revocation or forward/backward secrecy in distributed systems. For the design of future secure distributed computing systems, the challenge of anonymous and dynamic authentication must be addressed. More explicitly, there is a pressing need for a generalized framework to perform (a) robust authentication with (b) strong privacy guarantees, and (c) fine-grained context-aware access control mechanisms.

In this work, we introduce a design methodology to anonymously authenticate objects/entities in distributed systems. The major contributions are:

1) A novel cryptographic group anonymous authentication protocol (GAAP) that enables entities to be authenticated through their group attributes, while preserving the privacy of their individual identities.
2) An architectural support for GAAP - the proposed architecture and its implementation provide a convenient hardware add-on to enable the seamless integration of the protocol into existing device networks without major modifications to their topology or functionality.

Section II reviews the related works in this field that form the base upon which we propose the desired functionality and security criteria of GAAP. Sections III and IV present the system design flow of the trustworthy group anonymous authentication and its hardware architecture implementation, respectively. Section V concludes the paper with open problems and future work.

II. Related Work and Design Criteria

In this section, we briefly introduce the current research efforts trying to achieve group anonymous authentication (or some aspect of the problem). Their deficiencies in either functionality or security lead to the proposal of our GAAP scheme.

A. Related Works

A naive solution would be to have all the devices in a group use an identical ID as their individual identities. However, this suffers from many vulnerabilities. A compromised device can spoof other devices or be leveraged to make unlimited counterfeits, since they all share the same static secret. Harn [2], [3] first brought up the concept of group authentication using threshold secret sharing. This procedure allows the validation of an entire group of devices in one authentication step. However, due to the mathematical nature of its group tag reconstruction, all the devices' identities have to be disclosed, failing to provide the anonymity required in GAAP.

Anonymous authentication proposed by Camenisch [4], [5], has been widely adopted in many areas. The scheme, based on blind signature, has been applied to the e-cash protocol and the VOPRF algorithm in Tor networks [6]. However, since the authentication tags are generated by the edge devices themselves, this is hardly a trusted way to verify their group attributes. The group signature (GS) scheme by Chaum *et al.* allows any group member to sign a piece of data anonymously on behalf of the group [7], [8]. This scheme is now widely adopted in securing vehicular ad hoc networks (VANETs) [9]. The scheme relies on a key manager in the initial setup, leaving the devices' privacy susceptible to curious authorities. Furthermore, there are also challenges with dynamic membership removal and counterfeits from hijacked devices.

Verifiably Common Secret (VCS) encoding [10] was proposed by Schechter *et al*. In each authentication session, the verifier produces an encoded secret that can be decoded by any device of the same group. Therefore, when a device returns the decoded secret, it is indistinguishable from other group members. However, as with Chaum's group signature, this signature-based scheme also has a potential vulnerability to a curious authority (the certificate manager) and unlimited counterfeits on any hijacked device. These are critical security gaps, especially in heterogeneous distributed computing systems like SoCs and IoT networks, where a key can leak or a device can be hijacked [11].

B. Design Criteria

Based on these inadequacies, we define the desired functionality and security for a trustworthy *Group Anonymous Authentication Protocol* (GAAP) as follows:

Definition II.1. The **functionality** of GAAP should satisfy:

(i) ***Anonymity***: All devices in the same group should be indistinguishable to the verifier;
(ii) ***Group Attribute***: The verifier should be able to distinguish requests from different groups, without revealing the individual identities of the requesting devices;
(iii) ***Unlinkability***: The verifier should not be able to link several authentication requests/transactions to the same anonymous device;
(iv) ***Double-blindness***: In addition to remaining anonymous to the verifier, devices must be oblivious to the interactions between other devices and the verifier;
(v) ***Dynamic Group Membership***: A device should be able to belong to multiple groups. In other words, groups can overlap. New devices should be able to join groups and old devices should be able to leave.

Definition II.2. The **security** of GAAP should satisfy:

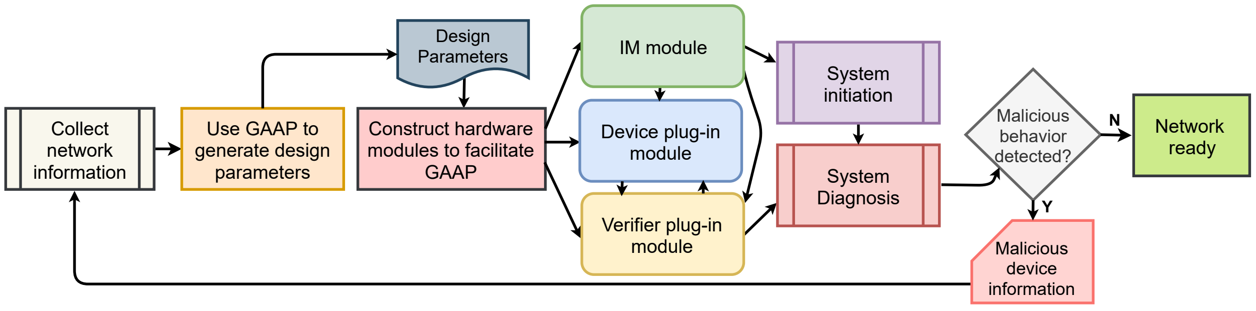
(i) ***Curious Authority-resistance***: A curious authority in the network, such as a verifier, a key distributor, or a certificate manager, should not be able to learn the individual identity of any device;
(ii) ***Impostor-resistance***: Any device in a group should not be able to spoof another. In addition, any number of dishonest devices working together should not be able to acquire more information than what they are legally granted;
(iii) ***Counterfeit-resistance***: The ability of attackers to introduce counterfeits by a hijacked device should be strictly restricted;
(iv) ***Eavesdrop-resistance***: A passive Man-in-the-Middle (MITM) should not be able to acquire any information by eavesdropping on the channel.

In the following two sections, we propose the design flow of a trustworthy group anonymous authentication system satisfying both the functionality and security criteria above.

III. Group Anonymous Authentication Protocol (GAAP) Design

A. Design Flow and Methodology

Figure 2 gives the design flow overview of the *Group Anonymous Authentication Protocol* (GAAP). The main steps of the design and deployment process are as follows:



Collect network information
Use GAAP to generate design parameters
Design Parameters
Construct hardware modules to facilitate GAAP
IM module
Device plug-in module
Verifier plug-in module
System initiation
System Diagnosis
Malicious behavior detected?
N
Y
Network ready
Malicious device information

Fig. 2: Group Anonymous Authentication Protocol (GAAP) Design Flow.

1. Collect the target network attributes, including the groups and the group members;
2. Generate system design parameters based on the collected attributes;
3. Instantiate set of GAAP hardware modules using the design parameters, and install the modules on the devices and the verifier;
4. Initialize physical system by having each device fetch its group identity certificates from the verifier. The system will diagnose to detect any malicious device behavior and update the network information accordingly. If no such behavior is detected, then the network is activated for GAAP.

Below, we expand each of the phases of the GAAP deployment process, i.e., algorithm, implementation or reconfiguration and activation phases.

B. Illustrative Demonstration

Before we present the mathematical definition and proof, we sketch out an analogy to illustrate the GAAP protocol's key characteristics.

Analogy III.1. A manager wants to give several groups of employees one-time access codes to different buildings, so that each employee in a certain group can access a corresponding building anonymously. For example, in group A with three employees, the manager prepares three different access codes for building x. The criteria governing access code assignment are:

a. For security reasons, the access codes cannot be left in an unsupervised room for the employees to pick up freely;
b. There should be no collision in the picking of the codes among the three employees;
c. The manager should not know any employee's code selection;
d. An employee should not know his colleagues' codes.

Thus, any of the employees can enter building x with their access code, without leaking their personal identification.

Such a scheme can be implemented with the procedure illustrated in Fig. 3.

① Initial setup:

a) There are three types of papers $(blue, green, red)$ and three corresponding solutions $(\alpha_b, \alpha_g, \alpha_r)$. When a solution is applied to its paired type of paper, it does no harm to it. However, when it is applied to other types, it permanently wipes out all the content from them. Access codes a, b, c are written to the $blue, green$, and red papers respectively for multiple copies, one copy of each for each employee;

b) In addition, each of the 3 employees has a special colorless solution denoted by $\beta_1, \beta_2, \beta_3$, which can temporarily erase the content from any of the paper types. Reapplying this solution will restore the content. However, β_i cannot restore the content masked by β_j if $i \neq j$.

c) Lastly, there is a solution γ, which, when mixed with α, changes color to a random bijection mapping to the set $\{blue, green, red\}$, which is unknown to the manager.

② The three employees are given the three solutions, and each picks one solution. At this point, no one has information about the access codes (a, b, c). In Fig. 3, employee 1 picks α_b, acquiring access code a; employee 2 picks

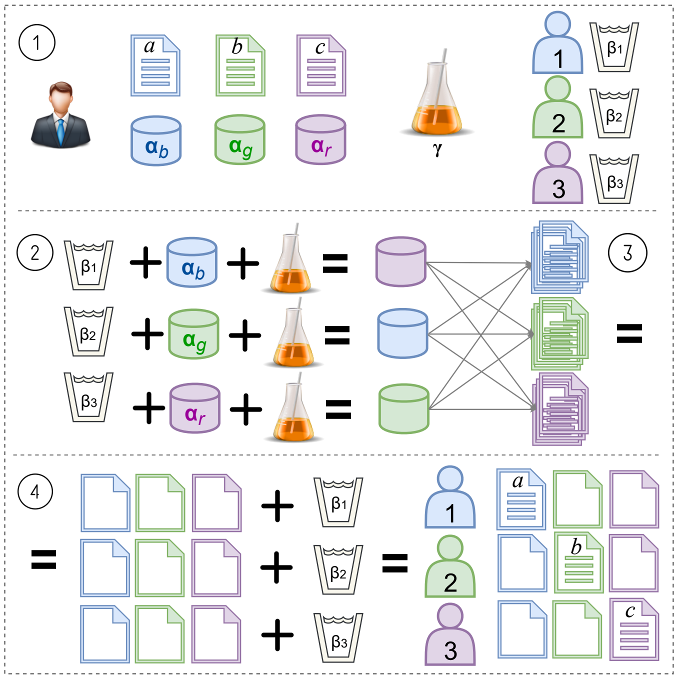


Fig. 3: A 4-step access code blind-fetching protocol. With this protocol, each employee can only acquire his/her own access code, and will remain completely unaware of other codes. The manager also does not know the code selection of the employees.

α_g, acquiring access code b, and employee 3 gets α_r for code c. Each employee's solution β is mixed with the selected α respectively, and then with solution γ. The resulting solution colors are a bijection of the original color set. For example, in Fig. 3, $\alpha_b + \beta_1$ is converted to red, $\alpha_g + \beta_2$ to blue, and $\alpha_r + \beta_3$ to green.

③ The manager looks at the three new solutions and has no knowledge of what their original colors were. Thus he also does not know about the employees' access code selection. However, the manager can tell if they have made a collision-free selection. He applies each of the solutions to a set of access code papers $(green, blue, red)$ with a, b, c on them. All the papers are now wiped out, some temporarily, and some permanently.

④ When each employee gets back their set of blank papers, they apply their own β solution to those papers. Only the previously picked color paper with the correct access code can be restored. The other two papers will remain garbled. The employee can now use their assigned code (employee 1 gets a, 2 with b, 3 with c) to access building x without being specifically identified. □

Similar procedures can be applied to other groups or other departments. This illustrative analogy already satisfies some of the important criteria in DEFINITION II.1 and II.2, including *Anonymity*, *Group Attribute*, *Double-blindness*, *Curious Authority-resistance*, *Impostor-resistance*, and *Eavesdrop-resistance*. A more detailed and rigorous mathematical formulation of the protocol now follows.

C. The Group Anonymous Authentication Protocol

We adopt tag matching as the authentication approach in the proposed protocol. Compared with signature-based schemes, tag matching can more effectively restrict counterfeits on hijacked devices. In such a protocol, a device proves its legitimacy by showing the verifier a one-time authentication tag with its group attribute, which has been previously acquired from the verifier itself in a double-blind manner.

We first define the following functions.

- **TagGen(**n, m**)**: for a group of n devices where each device can be authenticated m times, the verifier uses this function to generate nm tags.
- **IndexSel(**$\{j\}, \{i\}$**)**: all devices in a group use this function to create a bijection mapping from the device IDs to the tag indexes. The output of this function is a set $\{c_j\}$, where $c_j = i$, meaning device j has chosen the i^{th} tag.
- **ColnChk(**$\{z\}$**)**: this function checks if there are duplicates in the extensional set $\{z\}$.
- **Enc(**x, pk_j**)**: the verifier uses this function to encrypt a message x with the public key pk_j of device j.
- **Dec(**y, sk_j**)**: device j uses this function to decrypt y with its private key sk_j.

In addition to the above functions, the protocol uses:

- **Initializer Module (IM)**: its only task is to generate and distribute random numbers. It has no other functional capability and cannot participate in any other activity.
- $\oplus$: an involution operator such as xor.

Protocol III.1. Let us assume that there are w devices and u groups in a distributed system. For an arbitrary group g, there are n devices.

1) For the group g, the server uses **TagGen(**n, m**)** to generate nm arbitrary authentication tags t_i stored in set T. There are n devices $\{dev_j\}$ indexed by $j \in \{0, 1, \cdots, n-1\}$. Each device fetches m tags to enable m authentication sessions. Every device has a public-private key pair: $\{pk_j, sk_j\}$;
2) In the 1st round of tag acquisition, the server arbitrarily takes n tags out of the nm tags, and indexes them by $i \in \{0, 1, 2, \cdots, n-1\}$ as $\{t_0, t_1, t_2, \cdots, t_i, \cdots, t_{n-1}\}$;
3) The n devices come to an agreement over the index i selection with **IndexSel(**$\{j\}, \{i\}$**)** $= \{c_j\}$. Each device's choice is $c_j = i$, meaning that the device dev_j plans to acquire t_{c_j}. This agreement should be performed in a collision-free manner. (Any double-dealing in the agreement will be spotted in step 6 by the verifier);
4) An Initializer Module (IM) generates n random vectors $\{r_0, r_1, r_2, \cdots, r_{n-1}\}$, and a random number $d \in \{0, 1, 2, \cdots, n-1\}$. The IM sends d and r_d to all the devices. It also sends all the random vectors $\{r_0, r_1, r_2, \cdots, r_{n-1}\}$ to the verifier. The transmission can be protected against MITM eavesdropping simply by using the public key systems of the verifier and devices;
5) Each device computes
$$e_j = c_j + d \ (\text{mod } n), \tag{1}$$
and sends e_j to the verifier. e_j leaks zero-knowledge of c_j since the verifier has no knowledge of d;
6) The verifier checks if there are any index selection collisions using the function **ColnChk(**$\{e_j\}$**)**, where $j \in \{0, 1, \cdots, n-1\}$. If any device fools the index selection agreement function **IndexSel()** in step (3), it will be detected with probability of 1 without nullifying the tag selection process. If no collision exists, the verifier proceeds;
7) For any device dev_j's e_j, the verifier computes
$$f_i = t_i \oplus r_{e_j - i \ (\text{mod } n)} \tag{2}$$
for all $i \in \{0, 1, \cdots, n-1\}$, and stores the results in the set of $\{f_i\}_{dev_j}$ whose cardinality is n;
8) The verifier uses each device's public key pk_j to compute **Enc(**$\{f_i\}_{dev_j}, pk_j$**)**, and sends the results to dev_j;
9) When device dev_j receives its tag set, it uses **Dec(Enc(**$\{f_i\}_{dev_j}$**,** pk_j**),** sk_j**)** to decrypt and retrieve $\{f_i\}$. Then by the previously received r_d from the IM, dev_j computes its selected authentication tag by
$$t_{c_j} = f_{c_j} \oplus r_d, \tag{3}$$
and it has zero-knowledge of the other tags;
10) All the participants repeat the steps above for another $(m-1)$ rounds allowing each device to acquire m tags. In each round, the IM generates a new set of random vectors $\{r_0, r_1, r_2, \cdots, r_{n-1}\}$ and a new d;
11) During the authentication itself, a device shows one of its tags to the verifier to prove that it belongs to group

g. If that tag matches a tag in the verifier's T set, the device is successfully authenticated. Then, the device can legitimately request privileges assigned to its group without revealing its individual identity. ■

Remark III.1. We can see the following parallels correlating the building access code distribution analogy and Protocol III.1:

- d generated by the IM is equivalent to the solution γ;
- c_j is equivalent to the solution α;
- The device's public-private key pair is equivalent to the solution β ;
- PROTOCOL III.1 (7) is equivalent to the step (3) in the "access code" analogy, which erases the selected paper temporarily, and destroys all other papers permanently. □

Remark III.2. We now discuss how Protocol III.1 satisfies DEFINITIONS II.1 and II.2.

Since the selection of authentication tags is obscured by the random number d (updated in each round), which is unknown to the verifier, the verifier cannot link any tag(s) to a device. Thus both *Anonymity* and *Unlinkability* are satisfied, and the protocol is therefore also *Curious Authority-resistant*.

By keeping a record of each group's tag pool, the verifier is able to tell the *Group Attribute* of each authentication request. As for the *Flexible Group Membership*, a single device can possess memberships in multiple groups by legally fetching the authentication tags of those groups. A device can join a group by repeating Protocol III.1. A device is also able to leave in an anonymous manner by returning all its tags to the verifier, who remains unaware of the change.

Besides being anonymous to the verifier, steps 8 and 9 of PROTOCOL III.1 and [Eq. 2] also ensure that a device or a Man-in-the-Middle cannot acquire information of other devices' tags. Thus, the protocol achieves *Double-blindness* and has *Impostor-resistance* and *Eavesdrop-resistance*.

For any device, possessing m tags enables it to be authenticated up to m times. Consequently, even with device hijacking, the number of possible counterfeits is also strictly restricted, due to the tag matching mechanism. In contrast, in signature-based schemes, the counterfeits can be unlimited upon device hijacking or key leakage [11]. □

IV. GROUP ANONYMOUS AUTHENTICATION PROTOCOL (GAAP) IMPLEMENTATION

In this section, we present the GAAP hardware architecture details through an FPGA implementation illustration. As mentioned in Section I, because the hardware architecture is implemented as a standalone module, it enables convenient add-on deployment in existing connected device network systems. The add-on consists of three components: an initializer module, a plug-in for each device, and a plug-in for the verifier.

A. Initializer Module (IM)

The only task of the initializer module (IM) is to generate and distribute random numbers. It does not participate in any other activity. An IM consists of two sub-modules as shown in Fig. 4: a random number generator (RNG) and an encryption unit (ENC) to carry out the **Enc()** function with RSA/R-LWE.

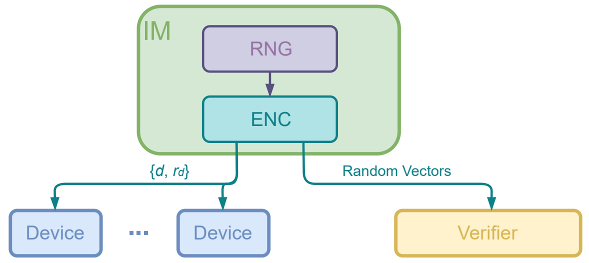


Fig. 4: The IM functions as described in Protocol III.1 step 4.

B. Authentication Plug-in for Devices

As shown in Fig. 5, the first task of a device's plug-in is to coordinate with other devices, in order to achieve the collision-free index selection agreement in Protocol III.1 step 3.

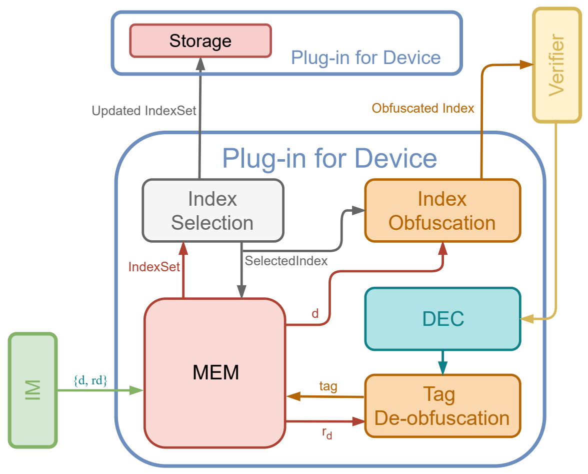


Fig. 5: The device plug-in and its interaction with other entities.

The actual algorithm carried out in the Index Selection module is:

```
IndexSet = [0 to n-1]
from device 0 to device n-1:
   SelectedIndex = random.choice(IndexSet)
   IndexSet = IndexSet.remove(SelectedIndex)
```

Algorithm 1: Index Selection Agreement

Then the SelectedIndex c_j is obfuscated by [Eq. 2] in Protocol III.1 step 5 before being sent to the verifier.

In Protocol III.1 step 9, on receiving the encrypted authentication tags, the DEC and Tag De-obfuscation modules are able to retrieve the targeted tag t_{c_j} using the **Dec()** function with a RSA or R-LWE public-key scheme and de-obfuscation [Eq. 3].

C. Authentication Plug-in for the Verifier

In the verifier plug-in, shown in Fig. 6, the Tag Generator module is another RNG, generating nm authentication tags as in Protocol III.1 steps 1 and 2.

On receiving the obfuscated index selections from the devices, the Collision Check module functions as Protocol III.1

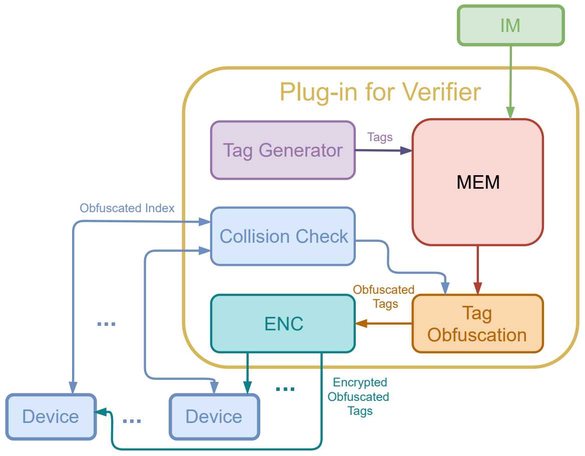


Fig. 6: The verifier plug-in and its interaction with other entities.

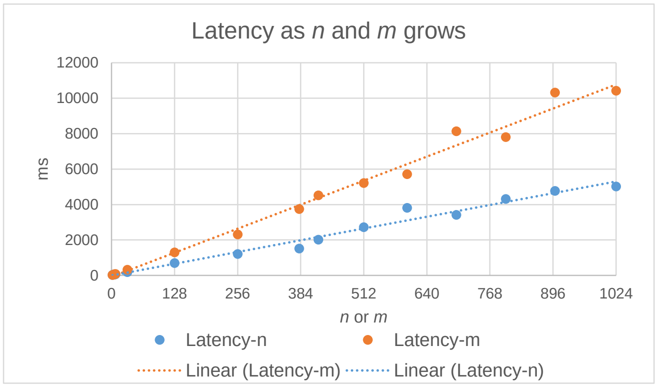


Fig. 7: As m (fetching round) grows, the latency is almost m times the single round latency since each round is independent. As n grows, the latency grows with a smaller slope because many operations within one round can be parallelized.

step 6. Since the obfuscated index selection is a bijection to the original index set, we are able to adopt an efficient collision check algorithm. First, each obfuscated index is assigned an extra FlagBit. The FlagBit is initialized to 0, indicating this vector has not been visited yet; it flips to 1 when visited. Then the Collision Check module traverses the obfuscated index set and uses each element as a pointer to visit another. If any element is visited more than once, a collision is detected.

```
ObIndexSet = [n obfuscated indexes]
for i in range(0, n):
    if ObIndexSet[ObIndexSet[i]].FlagBit == 0:
        ObIndexSet[ObIndexSet[i]].FlagBit = 1
    else:
        report collision on ObIndexSet[i]
```

Algorithm 2: Collision Check with time complexity $O(n)$

With the previously received n random vectors, the Tag Obfuscation module performs [Eq. 2] in Protocol III.1 step 7. [Eq. 2] is a critical operation to ensure the delivery of t_{c_j} and the concealment of other tags. The obfuscated tags are then encrypted by the ENC module before being sent back to the devices, as in Protocol III.1 step 8.

The latency of a complete authentication tag fetching is affected by both n, the number of devices, and m, the number of fetching rounds. The trends are shown by Fig. 7.

When nm grows larger, tag fetching can take longer. However, the task is merely a one-time initialization. Once it is done, the authentication procedure is simple and fast.

V. Conclusion

A secure group anonymous authentication protocol (GAAP) is introduced in this work. It enables a device to be authenticated without revealing its individual identity. In a distributed computing system setting, using GAAP, devices can be verified using their group attributes. Unique or specific identities tied to physical devices can be concealed, and end-users' privacy can be preserved. Moreover, this protocol provides strong security guarantees against various attacks. Curious authorities or dishonest group members cannot breach the anonymity of devices. The creation of counterfeits is also strictly restricted.

Although the proposed protocol satisfies both Definitions II.1 and II.2, improvements to the technique can still be made. For example, can the initializer module (IM) – which we currently assume to be a trusted third party – be removed from the protocol without loss in functionality or weakening security? Addressing this challenge will also eliminate the hazard of the IM being compromised, which is beyond the scope of this paper and remains an open problem.

References

[1] P. N. Brown, H. Borowski, and J. R. Marden, "Security against impersonation attacks in distributed systems," *IEEE Transactions on Control of Network Systems*, 2018.
[2] L. Harn, "Group authentication," *IEEE Transactions on computers*, 2013.
[3] L. Harn and C. Lin, "An efficient group authentication for group communications," *arXiv preprint*, 2013.
[4] J. Camenisch, S. Hohenberger, and A. Lysyanskaya, "Compact e-cash," *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2005.
[5] J. Camenisch, S. Hohenberger, and M. Kohlweiss, "How to win the clonewars: efficient periodic n-times anonymous authentication," *Proceedings of the 13th ACM conference on Computer and communications security*, 2006.
[6] A. Davidson, I. Goldberg, N. Sullivan, G. Tankersley, and F. Valsorda, "Privacy pass: Bypassing internet challenges anonymously," *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 3, pp. 164–180, 2018.
[7] D. Chaum and E. Van Heyst, "Group signatures," in *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer, 1991, pp. 257–265.
[8] J. Camenisch and M. Stadler, "Efficient group signature schemes for large groups," in *Annual International Cryptology Conference*. Springer, 1997, pp. 410–424.
[9] Y. Sun, Z. Feng, Q. Hu, and J. Su, "An efficient distributed key management scheme for group-signature based anonymous authentication in vanet," *Security and Communication Networks*, vol. 5, no. 1, pp. 79–86, 2012.
[10] S. Schechter, T. Parnell, and A. Hartemink, "Anonymous authentication of membership in dynamic groups," in *International Conference on Financial Cryptography*. Springer, 1999, pp. 184–195.
[11] S. Khandelwal. Millions of iot devices using same hard-coded crypto keys. [Online]. Available: thehackernews.com/2015/11/iot-device-crypto-keys.html