Anonymous Broadcast Authentication with Logarithmic-Order Ciphertexts from DLP or LWE

Yoshinori Aono[1,2] and Junji Shikata[3,1]

[1] Institute of Advanced Sciences, Yokohama National University 79-5 Tokiwadai, Hodogaya-ku, Yokohama 240-8501, Japan.
[2] NICT, 4-2-1, Nukui-Kitamachi, Koganei, Tokyo, Japan.
[3] Graduate School of Environment and Information Sciences, Yokohama National University, 79-7 Tokiwadai, Hodogaya-ku, Yokohama 240-8501, Japan

Abstract. We propose an anonymous broadcast authentication (ABA) scheme to simultaneously control massive numbers of devices in practical resources. As a theoretical foundation, we find a barrier in constructing an ABA scheme that can control numerous devices: a trilemma between (i) security, (ii) ciphertext length, and (iii) freedom of target device selection. Therefore, we propose ABAs with ciphertext sizes of $O(\log N)$, where N is the number of target devices and impose a certain restriction on (iii). We provide an ABA template and instantiate it into specific schemes from the decisional Diffie-Hellman problem or the learning with errors problem. Further, we provide example parameters and resource consumption of space and time.

Keywords: Anonymous broadcast authentication · IoT Network · Discrete logarithm problem · Learning with errors problem

1 Introduction

The anonymous broadcast authentication (ABA) framework [32] is a one-way communication from a central server to multiple resource-limited devices. The server broadcasts a command to control a set of devices. The following conditions (1) and (2) are the minimum desired specifications for correctness.

(1) A message from the server includes information on the IDs of the target devices and control commands. Each device that receives the message either executes the command if included in the target set or does nothing if otherwise.
(2) The received message has integrity and authenticity.

Two additional security notions (3) and (4) should also be satisfied.

(3) Unforgeability: In a situation in which secret information about some devices is leaked, an adversarial entity cannot forge a legitimate command with the information.
(4) Anonymity: Each device can detect whether it is a target but cannot determine whether another device is a target.

The application that we envision is sending emergency signals to reboot, shut down malware-infected devices. Thus, we assume that the space required to send these signals and optional flags is small (a few bits). We expect the number of devices to be approximately 10^6 – 10^9 to control

them all within a base wireless area (several square kilometers) simultaneously in the Internet of Things (IoT) of 5G or the network beyond it. The entire process of command generation in a central server, communication, and authentication in the target devices must also be completed within a few seconds in resource-limited devices for a fast response to an emergency.

For applications, the command should be encrypted to satisfy conditions (3) and (4). However, there are several barriers regarding ciphertext length. Assuming the atomic model [19, 20], an ABA that has anonymity must have $\Omega(n)$ ciphertext length, where n is the number of target or joined devices.

it depends on the security requirement of anonymity. A similar bound is derived simply from Shannon's coding theorem because condition (1) requires that the information amount contained in the ciphertext exceeds the number of participant devices N. This bound holds under the assumption that a set of target devices is randomly selected from a family of subsets of N devices.

These observations deduce the following trilemma: In ABAs, (i) security (anonymity), (ii) short ciphertext length, and (iii) freedom of target devices selection are not simultaneously satisfied. Therefore, for practical use, we propose an ABA with a ciphertext size of $O(\log N)$ by imposing a certain restriction on (iii). Precisely, our ABA protocol has device IDs represented by a vector $(\mathsf{id}_1, \ldots, \mathsf{id}_K)$ where $\mathsf{id}_j \in [N_j] := \{1, 2, \ldots, N_j\}$ and ciphertext length $O(\sum N_j)$. It can control $\prod N_j$ devices, which is an exponential number of the ciphertext length.

First, we construct an ABA template and provide instantiations from the decisional Diffie–Hellman (DDH) or the learning with errors (LWE) problems. In the LWE-based construction, using a parameter set controlling 10^6 – 10^9 devices with 128-bit security, the ciphertext length is approximately 360KB – 1MB and the expected processing timing of the verification in target devices is a few seconds in ARM Cortex-M4 processors; details are provided in Section 7.2 and Tables 2 and 3.

1.1 Related Work

Broadcast encryption (BE) The notion of BE is considerably similar to that of ABA. At the formal definition level, the notion of ABA is equivalent to that of the private key BE (prBE) in [27, Def. 3.1]. However, security notions are slightly different since the considered applications are different. Of note, the ABA is an authentication-oriented analog of anonymous BE (ANOBE).

The anonymity notion in the BE framework, which corresponds to the weak anonymity in the ABA framework, was introduced by Barth et al. [4]. Afterward, Benoît et al. [23] proposed an efficient scheme from the DDH assumption over groups with the notion of ANOBE, and [16] presented the lattice interpretation. They allowed the freedom of target device selection and thus required a linear ciphertext size, precisely, $O(N-r)$, where N and r denote the numbers of joined and revoked devices, respectively. Their constructions have complete freedom of target device selection, and the ciphertext sizes cannot be smaller than $O(N), O(r)$ or $O(N-r)$ that touch the theoretical asymptotic limit. Fazio et al. [11] proposed a public key BE with outsider anonymity and CPA/CCA security and achieved a log-order ciphertext size $O(r \log(N/r))$ that is linear in r.

For practical use on IoT devices, decryption timing should be considered. Efficient implementations of BE have been realized in several existing works. [8] surveyed of several pairing-based BE systems and implemented them in the same environment on a standard laptop. Notably, sublinear ciphertext sized BEs with some functionalities [7, 14] can achieve a decryption timing of less than one second, even for $N = 10^6$.

Theoretically, there are two major issues if BE algorithms are imported directly into our ABA. First, constructing a practical scheme with short ciphertext ($o(N)$ [bit]) while maintaining reasonable anonymity is challenging in BE and its variants. It is nontrivial to import an existing scheme into the ABA framework and prove anonymity. Second, a transformation technique to add unforgeability is useful, as explained below. Interpreting authentication results as the transmission of a one bit message, the ABA framework can be considered a prBE with additional functionalities. Therefore, the proposed ABA scheme can also be considered as a new variant of short ciphertext prBE.

Atomic model in the BE framework This model assumes that the server broadcasts a sequence of ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$ that encrypts a control command. Each device j then attempts decrypting each ct_i using its key dk_j.

The sequence length ℓ in this model was studied in the prBE by Kiaias-Samari [19] and in ABA by Kobayashi et al. [20]. They showed that $\ell \geq N$ if an ABA controlling N devices has anonymity. It deduces the total bit-lengths of ciphertexts is $\Omega(N \cdot \lambda)$, where λ is the security parameter. The bounds can be relaxed to $\ell \geq |\mathcal{S}|$ and $\Omega(|\mathcal{S}| \cdot \lambda)$, respectively if weak anonymity is assumed instead of the anonymity where $\mathcal{S} \subset [N]$ is the set of target devices. Concrete constructions achieving the bounds in both cases have been known in [32, 20].

Transformation to add unforgeability It can be performed using a technique that converts a weak security prBE to a CCA1 secure prBE. The simplest transformation should be the addition of a signature to the broadcasting ciphertext, as in [22]. Considering the similarity between the unforgeability of ABA and the CCA1 security of prBE, the proof of anonymity in our ABA is straightforward.

Infeasibility of naïve atomic ABA systems Consider atomic type ABAs constructed from a standard encryption and a signature scheme. We provide a rough estimation of ciphertexts to be transmitted. For example, a standard (resp. structured) lattice-based encryption needs tens of kilobytes (resp. half a kilobyte) of ciphertext length. FrodoKEM [12] and FALCON [10] provided good examples of sizes after optimization. A system for controlling $N = 10^6$ devices requires a gigabytes for one command ciphertext, which is too large to process on a low-resource device. Thus, an ABA with short ciphertexts that can control millions of devices is required.

1.2 Our Contributions

We propose a template ABA construction of short ciphertext and provide instantiations from DDH or LWE. Our major contributions and improvements over the CANS proceeding's version [2] are as follows.

Design rationale of our scheme From the viewpoint of the considered limitations and the linear lower bound of ciphertext length, our design rationale is organized as follows. Let At and An represent that an ABA is in the atomic model and has anonymity, respectively. Let LB represent that an ABA has $\Omega(N)$ or $\Omega(|\mathcal{S}|)$ ciphertext length on average over the selection of target sets and messages. Kobayashi et al.'s [20] result can then be described as follows:

$$[At \text{ AND } An] \Rightarrow LB.$$

In addition, we denote F as the complete freedom of target device selection, i.e., any $\mathcal{S} \subset [N]$ can be selected as a target device set, and assume that it is randomly chosen from $2^{[N]}$. According

to Shannon's coding theorem, the ciphertext must be longer than N bits on average because the broadcasting ciphertext entropy exceeds N bits. Thus, we obtain $F \Rightarrow LB$ and deduce the following relation

$$\neg LB \Rightarrow \neg F \text{ AND } [\neg At \text{ OR } \neg An].$$

This relation is interpreted as that an ABA with short ciphertexts must restrict either F, At, or An conditions. Our construction satisfies $\neg F$, $\neg At$, and "near" anonymity which we will detail in Section 6.3. We emphasize that it is on the feasible area's borderline.

We explain why we restrict the strength of anonymity by introducing a new notion of anonymity which we named it single anonymity or shortly SA; see, Section 6.2.

The notion SA is about information leakage on $\mathsf{id}' \overset{?}{\in} \mathcal{S}$, i.e., whether id' is in the target set, from other patterns $\{\mathsf{id} \overset{?}{\in} \mathcal{S}\}_{\mathsf{id}}$. We prove that $\neg F \Rightarrow \neg SA$. Thus, combined with $F \Rightarrow LB$, we conclude that any short ciphertext ABA inherently lacks SA. Therefore, we do not investigate the anonymity perfectly in the sense of the abovementioned An.

Of note, SA is slightly tight for practical situations. One of our future works is to investigate the relationship between the restriction of target device selection and the strength of anonymity. In addition, the relationship between the original t-anonymity (4) and SA is unclear.

Base atomic type ABA template We construct it from a Vernam-styled multirecipient encryption (MRE), which is a fundamental tool with information-theoretic security. Technically, the Vernam-styled communication protocol cannot be secure if the server sends ciphertexts with the same secret key several times. To address this issue, we transform MRE to a computationally secure ABA using a template function f_{prm} based on Kurosawa et al.'s [21] technique. We then instantiate the ABA to a practical protocol by using a standard and elliptic curve discrete logarithm-styled function, or an LWE-styled function. These template construction are within the atomic model that sends M ($\geq N$) ciphertexts to N devices. Although the discrete logarithm constructions are not secure against a large-scale quantum computer, the verification speed is much faster than that of lattice construction. Of note, this is essentially similar to the concatenation of the naïve multirecipient encryption in [6, Sect. 1.3].

ABA with logarithmic-order ciphertext length Through ABA concatenation, we develop an ABA template with short ciphertext length. However, the concatenated ABA does not have anonymity. Thus, we propose a modification based on Agrawal et al.'s inner-product encryption [1] to add anonymity in a limited sense. Finally, we add the unforgeability by adding a post quantum signature.

Each device is indexed by a vector $(i_1, \ldots, i_K)$ where $i_j \in [N_j] := \{1, 2, \ldots, N_j\}$ and each N_j is the size of each coordinate set as public parameters. The target set is defined by a sequence of sets $S_j \subset [N_j]$, and a device $(i_1, \ldots, i_K)$ is a target if $i_j \in S_j$ for all j. The length of ciphertext is $O(N_1 + \cdots + N_K)$. A trade-off between the ciphertext length and the flexibility of target sets can be considered by varying N_j. For instance, $K = 1$ corresponds to an atomic ABA. Meanwhile, for $K \geq 2$ and setting all N_j equivalent, it derives an ABA controlling N devices by $O(K \cdot N^{1/K})$ [bit] ciphertext. In particular, setting $N_j = 2$ for all j, it yields an ABA controlling $N = 2^K$ devices with $O(K) = O(\log N)$ [bit] ciphertext.

Improvement over the conference version [2] Based on the template construction, we add the instantiation from the discrete logarithm problem that achieves smaller space requirements and fast processing while it lacks quantum resiliency. Details are given in Section 7.1.

We add discussion of message security, which is defined as the hardness to distinguish two ciphertexts from different messages and the same target devices. In the lattice and the discrete logarithm constructions, the message securities are reduced to the decision LWE and DDH, respectively. Details are given in Section 5.3.

We provide a new template construction that achieves a shorter ciphertext length of $O(|\mathcal{S}|)$ by replacing anonymity with the weak anonymity. The concatenating construction achieves a shorter ciphertext length of $O(|\mathcal{S}_1| + \cdots + |\mathcal{S}_K|)$. However, its anonymity is considerable. Details and discussion are given in Section 8.

In response to questions in the ACNS conference [2], we have added discussions on how to continue manipulating devices when they move to an outside of the base wireless area, and how to disable malware. Because these topics are at another layer, we raise several open problems. Details are given in Section 1.4.

We also refine the parameters of LWE based construction by updating the analysis of error distribution. The ciphertext lengths are approximately 20 to 30% shorter than the conference version. Details are given in Section 7.2.

1.3 Data Size and Expected Timing

The parameters and expected performances are summarized in Table 1 in Section 7.1 (DLP-based construction) and Table 2 and 3 in Section 7.2 (LWE-based construction). In both settings, we assume controlling 2^{20} to 2^{30} devices by a few bit messages. We provide the sizes of a verification key and a control command and expected consuming cycles in verification in each target device assuming to include an ARM Cortex-M4 processor.

For the standard DLP setting (112-bit security), the verification keys and command ciphertexts are less than 50KB and 100KB, respectively. Meanwhile, for the ECDLP setting (128-bit security), they are reduced almost 10-fold. Notably, the processing of DLP-based construction is much slower than that of ECDLP because the DLP have to handle integers of thousands of bits. For the LWE setting (128, 192, or 256 bit classical security), the keys and ciphertexts are within 100K – 2MB, and the verification can be performed in a few hundred million cycles, which is expected to be within a few seconds using a processor works at 100MHz.

We remark that the verification is dominated by the computation of SHA-3 in signature verification, which requires 213 cycles/byte in the ARM Cortex-M4 processor [29]. We can dramatically speed up verification using a lightweight hash function. For instance, Chaskey [24] can work with seven cycles/byte in the same processor and reduce the number of cycles and expected timing to less than 15 million and 0.15 seconds, respectively.

Comparisons with existing works We briefly compare our proposing ABA schemes with existing ABA from the viewpoint on performance and security. Kobayashi et al. [20] proposed the MAC-based constructions satisfying the anonymity (resp. the weak-anonymity) whose command sizes are exactly $(N+2)\lambda$ (resp. $(|S|+2)\lambda$), which hits the non-asymptotic linear lower bound where λ is the security parameter. Watanabe et al. [33] proposed shorter command ABAs by relaxing the anonymity condition and employing improved Bloom filters; the command size is still linear $O(N \cdot \log_2(1/\mu))$ with respect to the number of devices. Here, μ is the false-positive rate, which they assumed is within $2^{-10} - 2^{-20}$. For reference, to control $N = 10^6$ devices within $\mu = 2^{-10}$, the command size is approximately 1.8 MB. Our logarithmic-order constructions achieve much shorter commands than the previous construction by sacrificing anonymity.

1.4 Issues on Practical Application Beyond Cryptography

According to questions in our talk at the CANS conference [2], we discuss issues related to disabling malware and handover raised in practical applications. Because these issues are outside of the cryptographic designs, we merely provide a brief overview.

The first issue is malware disabling. Our protocol sends a shutdown or reboot command. However, some malware may not be disabled after rebooting. Therefore, the device should be disconnected from the network for security. Notably, shutdown operations need to be carefully performed in flying vehicles such as drones. Thus, a mechanism for safe grounding in an emergency is also required.

In a situation in which a device is moving outside the base wireless area, a handover operation is needed. This mechanism is defined in the RRC protocol (Layer 3 of the control plane) in 5G and is independent of individual devices. Meanwhile, reboot and shutdown commands should be placed at a higher layer because the commands are sent to a group of devices controlled by the same operating system.

1.5 Paper Organization

Section 2 introduces basic definitions and notations of computational problems and ABA. The first half of this paper (Section 3–5) is devoted to the theoretical proposal. In Section 3, we present a Vernam-styled MRE that broadcasts an encrypted message only for target devices with information-theoretic security. In Section 4, we transform the MRE into a template of computationally secure ABA. Section 5 discusses its security (anonymity, unforgeability, message security and key security) and their instantiations from DLP and LWE settings.

The second half of this paper (Section 6–8) provides practical tweaks. In Section 6, we propose a template of ABA with short command ciphertexts by ABA concatenation. We also discuss its security in the sense of single anonymity. Section 7 presents concrete DLP- and LWE-based schemes. We also present concrete parameters for security strengths, ciphertext sizes, and expected timings over the ARM Cortex-M4 processor. In Section 8, we propose a shorter ABA construction by sacrificing anonymity and present some open problems about security. Finally, Section 9 presents concluding remarks, issues to be considered in practical use, and future work.

2 Preliminaries

$\mathbb{Z}$ and $\mathbb{N}$ are the set of integers and natural numbers, respectively. For $N \in \mathbb{N}$, let $[N] := \{1, \ldots, N\}$. Define $\mathbb{Z}_q := \{0, 1, \ldots, q-1\}$ where q is assumed to be an odd prime. $\mathbb{Z}_q^\times := \mathbb{Z}_q \setminus \{0\}$. The notation $a \xleftarrow{\$} A$ is the uniform sampling from a finite set A. Bold letters such as $\boldsymbol{c}$ represent a row vector, and its transpose $\boldsymbol{c}^T$ is a column vector. We use $\boldsymbol{u}_i$ to denote the i-th unit vector $(0, \ldots, 1, \ldots, 0)$; the dimension is omitted if it is clear from the context. For vectors and matrices, $||$ denotes concatenation. For two sets $\mathcal{S}_0, \mathcal{S}_1$ of target devices, $\mathcal{S}_0 \triangle \mathcal{S}_1$ is the symmetric difference $(\mathcal{S}_0 \setminus \mathcal{S}_1) \cup (\mathcal{S}_1 \setminus \mathcal{S}_0)$.

2.1 Computational Problems

The prime field DLP is the problem to find an integer z that satisfies $g^z \equiv a \pmod{q}$ from a given tuple (g, a, q) Here, q is a prime that defines the finite field $\mathbb{Z}_q$. g is a generator, i.e., any integer

$h \in \{1, \ldots, q-1\}$ can be written as $g^i \bmod q$ using some $i \in \mathbb{Z}$. a is an integer between 2 and $q-1$. The DDH assumption states it is hard to distinguish (g^a, g^b, g^{ab}) and (g^a, g^b, g^r) where a, b, r are randomly selected from $\mathbb{Z}_q$. The recommended parameter size is 2,048 bits which achieves 112-bit security [3, Sect. 5.5.1.1]. We remark that the digital signature based on DLP over $\mathbb{Z}_q$ has been removed from the FIPS 186-4 standard [25] because it has been updated to 186-5 [26] in February 2024. Our protocol based on DLP over $\mathbb{Z}_q$ is just for reference. On the other hand, other DLP variants are considered. In particular, elliptic curve instantiations are the most successful results; they can reduce communication costs while ensuring security. Curve25519 used in the Ed25519 signature has a 256-bit length public key and 512-bit length signatures that ensure 128-bit security using a compressed representation of a point on the curve.

The LWE problem [28] is a fundamental toolkit for constructing lattice-based schemes. For a dimension parameter n, a modulo q, and an error distribution χ, the decision LWE is defined by the problem of distinguishing the polynomial number of samples $\{(\boldsymbol{a}_i, \boldsymbol{a}_i \boldsymbol{s}^T + e_i)\}_{i=1,\ldots,m}$ and $\{(\boldsymbol{a}_i, u_i)\}_{i=1,\ldots,m}$, where $\boldsymbol{s}^T \in \mathbb{Z}_q^n$ is a random secret vector fixed at all samples. $\boldsymbol{a}_i$, e_i, u_i are random vectors from $\mathbb{Z}_q^n$, random errors from χ, and random elements from $\mathbb{Z}_q$ respectively. χ is typically the discrete Gaussian distribution $D_{\mathbb{Z},\sigma}$ whose density function defined over $\mathbb{Z}$ is

$$\Pr[X = x] \propto \exp(-x^2/2\sigma^2).$$

The goal of the search version of LWE is to recover $\boldsymbol{s}$ from legitimate samples $\{\boldsymbol{a}_i, \boldsymbol{a}_i \cdot \boldsymbol{s}^T + e_i\}_{i=1,\ldots,m}$. The polynomial time equivalence between decision and search is known [28]. We set the lattice parameter using Albrecht et al.'s [9] lattice estimator.

2.2 Anonymous Broadcast Authentication

We introduce the notion of ABA and its correctness, unforgeability, and anonymity by Watanabe et al. [32].

Definition 1. *An ABA is formally defined as a four-tuple functions* $\Pi = (\mathsf{Setup}, \mathsf{Join}, \mathsf{Auth}, \mathsf{Vrfy})$.
- $\mathsf{Setup}(1^\lambda, N, \mathcal{D}) \to \mathsf{ak}$: *An algorithm that outputs the authorization key* ak. 1^λ *is a security parameter,* N *is the maximum number of joined devices, and* $\mathcal{D}$ *is a family of sets* $\mathcal{S} \subset [N]$ *that can be used as the set of target devices.*
- $\mathsf{Join}(\mathsf{ak}, \mathsf{id}) \to \mathsf{vk}_\mathsf{id}$: *An algorithm that outputs a verification key* vk_id *embedded to the device* id.
- $\mathsf{Auth}(\mathsf{ak}, \mathsf{m}, \mathcal{S}) \to \mathsf{cmd}_\mathcal{S}$: *It outputs a command ciphertext that encrypts the information on the message* m *and the set* $\mathcal{S}$ *of target devices.*
- $\mathsf{Vrfy}(\mathsf{vk}_i, \mathsf{cmd}_\mathcal{S}) \to \mathsf{m}/\mathsf{reject}$: *It verifies the command ciphertext* $\mathsf{cmd}_\mathcal{S}$ *using the verification key* vk_id *and returns the message or* reject *if it is accepted or rejected, respectively.*

The abovementioned algorithms, except Vrfy, are assumed to be probabilistic polynomials. Of note, the family $\mathcal{D}$ is typically set to $2^{[N]}$ and is not presented explicitly in several early works, although we restrict the freedom in the choice of a subset in $[N]$ to construct a short ciphertext ABA.

Definition 2. *An ABA* Π *has correctness if for any fixed* $(1^\lambda, N, \mathcal{D})$, ak *allowed to be input,* $\mathcal{S} \in \mathcal{D}$*, and any* $\mathsf{m}, \mathsf{id} \in [N]$*, it holds that*

$$\Pr[\mathsf{Vrfy}(\mathsf{Join}(\mathsf{ak}, \mathsf{id}), \mathsf{Auth}(\mathsf{ak}, \mathsf{m}, \mathcal{S})) \to \mathsf{m}] = 1 - \mathsf{negl}(\lambda) \text{ if } \mathsf{id} \in \mathcal{S}, \text{ and}$$
$$\Pr[\mathsf{Vrfy}(\mathsf{Join}(\mathsf{ak}, \mathsf{id}), \mathsf{Auth}(\mathsf{ak}, \mathsf{m}, \mathcal{S})) \to \mathsf{reject}] = 1 - \mathsf{negl}(\lambda) \text{ if } \mathsf{id} \notin \mathcal{S}.$$

The probability is over random coins in Join *and* Auth *(and possibly* Vrfy*).*

Below are the game-based formal definitions of unforgeability and anonymity in a situation in which the receiver devices are colluded and can share their verification keys. Our notions which are essentially the same as those in the original work [32], although we explicitly express t, the number of corrupted devices in the security notion to consider a practical security in Section 6.3.

Definition 3. *(t-unforgeability [32]) Consider the game between a challenger* C *and an adversary* A.

0: C *and* A *share* $(1^\lambda, N, \mathcal{D})$ *and* C *runs* $\mathsf{Setup}(1^\lambda, N, \mathcal{D}) \to \mathsf{ak}$. *Let* $M_a = M_v = \phi$ *be messages used in the authentication and verification queries. In addition, let* $\mathcal{D} \subset [N]$ *and* $W \subset \mathcal{D}$ *be the sets of considered devices during the game, and colluded devices, respectively.* $\mathsf{flag} \in \{0, 1\}$ *is a variable that indicates whether the adversary is successful in forging.*

1: *(Key generation)* A *selects a set of considered devices* $\mathcal{D} \subset [N]$ *and sends it to* C. C *runs* $\mathsf{Join}(\mathsf{ak}, \mathsf{id}) \to \mathsf{vk}_{\mathsf{id}}$ *for all* $\mathsf{id} \in \mathcal{D}$.

2: *(Collusion query)* A *selects* $\mathsf{id} \in D$ *and sends it to* C. C *adds* id *to* W *and sends back* $\mathsf{vk}_{\mathsf{id}}$ *to* A. A *can repeat this step until* $|W| < t$.

3: *(Authentication query)* A *sends* $(\mathsf{m}, \mathcal{S})$ *to* C, *where the selection is limited within* $\mathcal{S} \subset D$ *and* $\mathsf{m} \notin M_v$. C *then runs* $\mathsf{Auth}(\mathsf{ak}, \mathsf{m}, \mathcal{S}) \to \mathsf{cmd}_{\mathcal{S}}$ *and returns it to* A.

4: *(Verification query)* A *generates a set* $(\mathsf{m}, \mathsf{id}, \mathsf{cmd}_S)$ *and send them to* C. C *runs* $\mathsf{Vrfy}(\mathsf{vk}_{\mathsf{id}}, \mathsf{cmd}_S)$ *and returns the output to* A. *If* $\mathsf{Vrfy}(\mathsf{vk}_{\mathsf{id}}, \mathsf{cmd}_S) = \mathsf{m}$, $\mathsf{id} \notin W$ *and* $\mathsf{m} \notin M_a$, *then it sets* $\mathsf{flag} = 1$ *else set* $\mathsf{flag} = 0$. *Add* m *to* M_v.

After repeating Steps 3 and 4, if there is a verification trial such that $\mathsf{flag} = 1$, *we define the output of the experiment* $\mathsf{Exp}^{\mathsf{CMA}}_{\Pi,\mathsf{A}}(\lambda, N, \ell)$ *to be* 1; *otherwise, it is* 0. *The advantage of* A *in the protocol* Π *is*

$$\mathsf{Adv}^{\mathsf{CMA}}_{\Pi,\mathsf{A}}(\lambda, N, \ell) := \Pr[\mathsf{Exp}^{\mathsf{CMA}}_{\Pi,\mathsf{A}}(\lambda, N, \ell) \to 1].$$

An ABA protocol Π *is said to be* t*-unforgeable if the advantage is a negligible function of* λ.

The above formal definition can be interpreted as follows. Suppose t devices are taken over and colluded. In a situation in which an attacker collects secret information in the devices, it cannot forge a legitimate command ciphertext that an uncolluded device will accept.. We construct our unforgeable ABA from a base ABA by adding a signature.

Next, we consider the following passive attack rather than the above active attack.

Definition 4. *[32] (t-anonymity) Consider the game between a challenger* C *and an adversary* A. *As in the definition of unforgeability, t indicates the number of colluded devices.*

0: C *and* A *share* $(1^\lambda, N, \mathcal{D})$ *and* C *runs* $\mathsf{Setup}(1^\lambda, N, \mathcal{D}) \to \mathsf{ak}$. *Let* $M_a = \phi$ *be the set of commands used in the authentication. In addition, let* $\mathcal{D} \subset [N]$ *and* $W \subset \mathcal{D}$ *be the set of considered devices during the game, and the set of colluded devices, respectively*

1,2: *The same as the Steps 1,2 in the unforgeability game (Definition 3)*

3: *(Authentication query)* A *selects a pair* $(\mathsf{m}, \mathcal{S}), \mathcal{S} \subset \mathcal{D}, \mathsf{m} \notin M_a$ *and sends it to* C. C *runs* $\mathsf{Auth}(\mathsf{ak}, \mathsf{m}, \mathcal{S}) \to \mathsf{cmd}_{\mathcal{S}}$ *and returns the output and adds* m *to* M_a.

4: *(Challenge query)* A *selects a command* $\mathsf{m} \notin M_a$ *and two sets of devices* $\mathcal{S}_0, \mathcal{S}_1$ *and sends them to* C. C *runs* $\mathsf{Auth}(\mathsf{ak}, \mathsf{m}, \mathcal{S}_b) \to \mathsf{cmd}_{S_b}$, *where* $b \in \{0, 1\}$ *is a random bit. Returns the ciphertext to* A. A *guesses* b' *for the random bit.*

We define the game output as 1 *if* $b = b'$*, i.e., the adversary succeeds in guessing, and* 0 *if otherwise. The advantage is*

$$\mathsf{Adv}^{\mathsf{ANO}}_{\Pi,\mathsf{A}}(\lambda, N, \ell) := \left| 2\Pr\left[\mathsf{Exp}^{\mathsf{ANO}}_{\Pi,\mathsf{A}}(\lambda, N, \ell)\right] - 1 \right|.$$

In Step 4, the considered sets $\mathcal{S}_0$ and $\mathcal{S}_1$ must satisfy

$$\mathcal{S}_d := (\mathcal{S}_0 \triangle \mathcal{S}_1) \cap W = \phi \tag{1}$$

to prevent a trivial distinguishing; if $\mathcal{S}_d \neq \phi$, A can check whether some $\mathsf{id} \in \mathcal{S}_d$ is in $\mathcal{S}_0$ via the decryption oracle.

Notably, that the condition (1) means that an adversary can choose sets of any sises. The notion of weak t-anonymity is defined by adding the condition

$$|\mathcal{S}_0| = |\mathcal{S}_1| \tag{2}$$

besides (1) in Step 4. In addition, the outsider t-anonymity is defined by replacing (1) with $(\mathcal{S}_0 \cup \mathcal{S}_1) \cap W = \phi$ in Step 4. It is slightly weaker than the weak t-anonymity, although there is no restriction on the size of sets [32].

3 Vernam-Styled Multirecipient Encryption with Information-Theoretic Security

As a base gadget to construct our ABA, we introduce simple multirecipient secret key encryption. It is a one-way protocol from a central server to N participant devices. The server packs a set of messages into one ciphertext and broadcasts it to the devices. Each device decrypts the ciphertext with its key. It has information-theoretic security on messages, i.e., each device i can recover the i-th message m_i but can gain no information on the messages m_j $(j \neq i)$ to the other devices.

Definition 5. *A multirecipient encryption (MRE) is formally defined as a three-tuple of functions* $\mathsf{MRE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$.
• $\mathsf{MRE.KeyGen}(N, \mathsf{pp}) \to (\mathsf{ek}, \mathsf{dk}_1, \ldots, \mathsf{dk}_N)$*: It outputs an encryption key* ek *and decryption keys* $\{\mathsf{dk}_i\}_{i\in[N]}$.
• $\mathsf{MRE.Enc}(\mathsf{ek}, \{m_i\}_{i\in S}) \to \mathsf{ct}$*: S is the set of target devices to which messages are sent. m_i is a message sent to the i-th device; the server can take different m_i for each i. An encryption algorithm outputs* ct *to be broadcast.*
• $\mathsf{MRE.Dec}(\mathsf{dk}_i, \mathsf{ct}) \to m'_i$*: A decryption algorithm that recovers a message in the i-th device from* ct *using its secret key* dk_i.

Construction The public parameter $\mathsf{pp} = (M, q)$ is the pair of a vector dimension and a prime modulus. The decryption keys are randomly generated independent column vectors $\mathsf{dk}_i^T \in \mathbb{Z}_q^M$. The encryption key is the set $\mathsf{ek} = \{\mathsf{dk}_1^T, \ldots, \mathsf{dk}_N^T\}$. Each participant device i has pp and dk_i^T. For a set of target devices $\mathcal{S} \subset [N]$ and a set of messages $\{m_i\}_{i\in S}$ (where $m_i \in \mathbb{Z}_q^\times$), the ciphertext ct is a randomly chosen vector in $\mathbb{Z}_q^M$ that satisfies $\mathsf{ct} \cdot \mathsf{dk}_i^T \equiv m_i \pmod q$ for all $i \in S$. Decryption at device i is inner product computation $\mathsf{ct} \cdot \mathsf{dk}_i^T \pmod q$. Thus, correctness is immediate.

Since each coordinate is of $\mathbb{Z}_q^M$, the sizes of the encryption key, decryption key, and ciphertext are $NM \log_2 q$, $M \log_2 q$, and $M \log_2 q$ (in bits), respectively.

Information-theoretic security Suppose the situation in which the devices $1, 2, \ldots, t$ are colluded and an attacker wants to recover message m_{t+1} of the device $t+1$ from ct using leaked keys $\mathsf{dk}_1^T, \ldots, \mathsf{dk}_t^T$. In this case, the attacker can only know the fact that dk_{t+1} is independent of $\mathsf{dk}_1^T, \ldots, \mathsf{dk}_t^T$. Suppose that the attacker guesses a vector $\boldsymbol{v}^T$ and $m'_{t+1} = \mathsf{ct} \cdot \boldsymbol{v}^T$ as candidates of dk_{t+1}^T and m_{t+1}, respectively. All vectors $\boldsymbol{v}^T, 2\boldsymbol{v}^T, \ldots, (q-1)\boldsymbol{v}^T \bmod q$ can also be candidates for secret keys with equal possibility. Thus, $\{m'_{t+1}, 2m'_{t+1}, \ldots, (q-1)m'_{t+1}\} = \mathbb{Z}_q^\times$ are also the set of candidates of the message with equal possibility. This means that the attacker gains no information about m_{t+1} from ct and colluded keys. A similar argument can prove the impossibility of forging ct that embeds a message to the device $t+1$.

Therefore, MRE can be used as an ABA with information-theoretic security in one-time broadcasting while any security under chosen-plaintext attacks and key reusing situations does not hold. Specifically, assume a situation in which the attacker can obtain ct corresponding to any chosen $\{m_i\}_{i \in S}$ and any S with fixed decryption keys. The attacker can recover dk_i^Ts by solving linear equations from a sufficient number of pairs of messages and ciphertexts. IN addition, there is no mechanism to determine whether a device is a target from the knowledge of a received ciphertext.

Remarks The following implementation technique can speed up encryption. Fixing the keys, the server precomputes vectors base_i and $\boldsymbol{z}_k$ so that $\mathsf{base}_i \cdot \mathsf{dk}_j^T = \delta_{ij} \bmod q$ and $\boldsymbol{z}_k \cdot \mathsf{dk}_j^T = 0 \bmod q$ for $\forall j, k$. The ciphertext for the message set $\{m_i\}_{i \in S}$ is then computed using random numbers r_i as

$$\mathsf{ct} = \sum_{i \in S} m_i \cdot \mathsf{base}_i + \sum_k r_k \cdot \boldsymbol{z}_k.$$

Thus, the encryption is computable in $O(N \cdot M)$ operations in $\mathbb{Z}_q$.

This construction can be regarded as a generalization of the concatenation of Vernam ciphers because in the situation where $N = M$ and all dk_i^T's are a multiple of i-th unit vector $\boldsymbol{u}_i^T$, the ciphertext $\mathsf{ct} = (c_1, \ldots, c_N)$ is the concatenation of c_i by which the i-th device can decrypt. We note the reason for setting dk_i^T independent vectors instead of $k_i \cdot \boldsymbol{u}_i^T$, where k_is are multiples. Consider a chosen plaintext attack in which an attacker can obtain a pair (m_i, ct) for an index i. The decryption key can then be easily found by a simple division where $\mathsf{dk}_i^T = k_i \cdot \boldsymbol{u}_i^T$. However, when dk_i's are independent, information-theoretic security can be ensured using vectors until M pairs of (m_i, ct) are obtained by the attacker. Using the M pairs, one can recover all dk_i^T's via the solution of simultaneous equations.

In the above construction, the length (dimension) of ciphertext is fixed. Modification to a variable length version is provided, and it will be used to construct a shorter ciphertext version of our protocol in Section 8. After the set of target devices is fixed, set the dimension of ct to $d = |\mathcal{S}| + 1$ and set the coordinates to satisfy $(\mathsf{ct}||\mathbf{0}) \cdot \mathsf{dk}_i^T \equiv m_i \pmod q$ for all $i \in \mathcal{S}$. The proof of information theoretic security is by the same way.

4 Template Construction of Base ABA

We transform the above information-theoretic MRE to ABA by adding a repeatable property. The main differences over the base MRE are as follows: (1) it broadcasts the same message to a selected

subset of devices and (2) its security is based on a computational problem assumption. We provide a template construction, instantiate it from DLP and LWE, and discuss its security.

Technically, the key idea of our transformation is Kurosawa et al.'s [21] of transforming a key predistribution scheme into a computationally secure multiple-usable broadcast encryption scheme.

4.1 Template

We present our base ABA template using a function $f_{\mathsf{prm}}(\boldsymbol{c}^T)$ defined over an r-dimensional column vector with a parameter prm. We assume that the function has the following "linear-like homomorphic" properties with respect to operations $(\circ, \otimes)$.

- For scalars a, b and vectors $\boldsymbol{x}, \boldsymbol{y}$, $f_{\mathsf{prm}}(a\boldsymbol{x}^T + b\boldsymbol{y}^T) = a \circ f_{\mathsf{prm}}(\boldsymbol{x}^T) \otimes b \circ f_{\mathsf{prm}}(\boldsymbol{y}^T)$.
- An inverse $f_{\mathsf{prm}}(\boldsymbol{x}^T)^{-1}$ of $f_{\mathsf{prm}}(\boldsymbol{x}^T)$ that satisfies $f_{\mathsf{prm}}(\boldsymbol{x}^T) \otimes f_{\mathsf{prm}}(\boldsymbol{x}^T)^{-1} = I$ (an unit) is easily computable.

These properties are used to compute a linear combination of vectors $f_{\mathsf{prm}}(\sum_{i=1}^{M} v_i \boldsymbol{y}_i^T)$ and its inverse to cancel out a mask in ciphertext $m \otimes f_{\mathsf{prm}}(\boldsymbol{x})$ in verification.

For instance, our DLP-based construction assumes $r = 1$, $\mathsf{prm} = g$, a generator of a finite field and define $f_{\mathsf{prm}}(x) = g^x$ for $x \in \mathbb{Z}_q$. The homomorphic property holds with $a \circ h = h^a$ and $a \otimes b = a \cdot b$. Concretely,

$$f_g(ax + by) = g^{ax+by} = g^{ax} \otimes g^{by} = a \circ f_g(x) \otimes b \circ f_g(y)$$

holds. Meanwhile, the ECDLP-based construction assumes $r = 1$ and $\mathsf{prm} = B$, where B is a base point with known order N. For $x \in [N-1]$, define $f_{\mathsf{prm}}(x) = xB$. Similar to the finite field situations, the operations are defined by $a \circ P = aP$ and $a \otimes b = a \cdot b \bmod N$.

In addition, the LWE-based construction is instantiated by setting $\mathsf{prm} = \boldsymbol{p} \in \mathbb{Z}_q^r$ and $f_{\boldsymbol{p}}(\boldsymbol{x}^T) = \boldsymbol{p}\boldsymbol{x}^T$. The homomorphic property holds with the definition $a \circ \boldsymbol{x} = a\boldsymbol{x} \bmod q$ and $\boldsymbol{x} \otimes \boldsymbol{y} = \boldsymbol{x} + \boldsymbol{y} \bmod q$ for an integer a and vectors $\boldsymbol{x}, \boldsymbol{y}$. Note that $\boldsymbol{p}$ in our construction has small coordinates from a discrete Gaussian though the above property holds for any vector.

Definition 6. *(*MRE*-based ABA template) Assume that the function* $f_{\mathsf{prm}}(\cdot)$ *has the above homomorphic property. A template of our ABA is defined as follows. Assume that* $\mathsf{pp} = (M, q)$ *in* MRE *is fixed from the security parameter* λ*.*

- $\mathsf{Setup}(1^\lambda, N, \mathcal{D}) \to \mathsf{ak}$: *Execute* $\mathsf{MRE.Setup}(N, \mathsf{pp} = (M, q)) \to (\mathsf{ek}, \{\mathsf{dk}_i^T\}) =: \mathsf{ak}$
- $\mathsf{Join}(\mathsf{ak}, \mathsf{id}) \to \mathsf{vk}_{\mathsf{id}}$: $\mathsf{vk}_{\mathsf{id}} := \mathsf{dk}_{\mathsf{id}}^T$
- $\mathsf{Auth}(\mathsf{ak}, \mathsf{m}, \mathcal{S}) \to \mathsf{cmd}_S$: *Randomly choose an* r*-dimensional column vector* $\boldsymbol{x}^T$ *from the domain of* f_{prm}*. Generate random small vectors* $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_N$ *from some distribution. Randomly choose a matrix* $CT \in \mathbb{Z}_q^{r \times M}$ *that satisfies* $CT \cdot \mathsf{dk}_i^T = \boldsymbol{x}^T + \boldsymbol{e}_i^T$ *for* $i \in \mathcal{S}$ *and* $CT \cdot \mathsf{dk}_i^T$ *is far from* $\boldsymbol{x}^T$ *for* $i \notin \mathcal{S}$*. Parse* CT *into the column vectors* $\mathsf{ct}_1^T, \ldots, \mathsf{ct}_M^T$*, encode them by* $F_i = f_{\mathsf{prm}}(\boldsymbol{c}\boldsymbol{t}_i^T)$*, and the command is* $\mathsf{cmd}_S = (\mathsf{m} \otimes f_{\mathsf{prm}}(\boldsymbol{x}), F_1, \ldots, F_M)$*.*
- $\mathsf{Vrfy}(\mathsf{vk}_{\mathsf{id}}, \mathsf{cmd}_S) \to \mathsf{m}/\mathsf{reject}$: *For the device's key* $\mathsf{dk}_i := (d_{i,1}, \ldots, d_{i,M})$*, compute* $f = d_{i,1} \circ F_1 \otimes \cdots \otimes d_{i,M} \circ F_M$*, and* $\mathsf{m}' = \mathsf{m} \otimes f_{\mathsf{prm}}(\boldsymbol{x}) \otimes f^{-1}$*.*

For correctness, it is necessary to constrain f_{prm} and error vectors. For a legitimate command and decryption key

$$\begin{aligned} &d_{i,1} \circ F_1 \otimes \cdots \otimes d_{i,M} \circ F_M \\ &= f_{\mathsf{prm}}(d_{i,1}\boldsymbol{c}\boldsymbol{t}_1^T + \cdots + d_{i,M}\boldsymbol{c}\boldsymbol{t}_M^T) = f_{\mathsf{prm}}(CT \cdot \mathsf{dk}_i) = f_{\mathsf{prm}}(\boldsymbol{x}^T + \boldsymbol{e}_i^T) \end{aligned} \tag{3}$$

holds. By the homomorphic property, $(\mathsf{m} \otimes f_{\mathsf{prm}}(\boldsymbol{x}^T)) \otimes f_{\mathsf{prm}}(\boldsymbol{x}^T + \boldsymbol{e}_i^T)^{-1} = \mathsf{m} \otimes f_{\mathsf{prm}}(\boldsymbol{e}_i^T)^{-1}$. Thus, f_{prm} and the error should be selected such that the optional decoding mechanism to recover m is performed efficiently.

In the (EC)DLP-based instantiations, we use $\boldsymbol{e}_i^T = 0$ for all i, and the verification function directly returns m. Meanwhile, in the LWE-based instantiation, $\boldsymbol{e}_i^T$ is a vector whose components are from $D_{\mathbb{Z},\sigma}$ and a rounding function is used to recover m.

For a nontarget device $i \notin \mathcal{S}$, the equation (3) becomes

$$\mathsf{m} \otimes f_{\mathsf{prm}}(\boldsymbol{x} - \boldsymbol{x}') \tag{4}$$

for another random $\boldsymbol{x}'$, which is far from $\boldsymbol{x}^T$. Although it does not help to recover the message, the result (4) is possibly included in the domain of legitimate commands. To prevent accidents, gimmicks should be used to separate the space of scalars into the legitimate commands and others. Our practical construction of LWE-based ABA considers such separation is described in Section 7.2.

Remark 1. Efficient sampling of matrix CT is possible via precomputation. Let $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{M-N}$ be independent vectors and each $\boldsymbol{c}_j$ satisfies $\boldsymbol{c}_j \cdot \mathsf{dk}_i^T = 0$ for all $i \in [N]$. Define the matrix C by $C^T = (\boldsymbol{c}_1^T, \ldots, \boldsymbol{c}_{M-N}^T)$. For any matrix $R \in \mathbb{Z}_q^{(M-N)\times M}$, $RC \cdot \mathsf{dk}_i^T = \boldsymbol{0}^T$ holds. Fixing the target vectors $\boldsymbol{y}_i^T = (y_{i,1}, \ldots, y_{i,r})^T$, which are set as $\boldsymbol{x}^T + \boldsymbol{e}_i^T$ or a vector far from $\boldsymbol{x}^T$, compute the initial vectors $\boldsymbol{b}_j$ so that $\boldsymbol{b}_j \cdot \mathsf{dk}_i^T = y_{i,j}$ by solving simultaneous equations. Let $B = (\boldsymbol{b}_1^T, \ldots, \boldsymbol{b}_r^T)^T$ and $CT = B + RC$ is the desired random matrix.

5 Security of ABA

First, we discuss anonymity (Section 5.1) and unforgeability (Section 5.2), which are considered in the original motivations of ABA [32]. We introduce a computational problem defined using f_{prm}, directly deduced from the anonymity game. In addition, we show that its DLP and LWE instantiations are reduced to the DDH or the LWE problems, respectively.

Second, we consider the message security (Section 5.3) and verification key security (Section 5.4). Message security is defined as the hardness to distinguish the command ciphertext without using the keys of target devices. For theoretical interest, we have proved this security in the DLP and LWE instantiations is reduced to the DDH and the decision LWE problems, respectively.

Verification key security is defined by the hardness of recovering the device's verification key vk_i or its alternative key $\mathsf{vk}_i'^T$, i.e., a key that $\mathsf{Vrfy}(\mathsf{vk}_i', \mathsf{cmd}_S)$ outputs the correct value with high probability of device i from other colluded keys $\{\mathsf{dk}_j^T\}$. We show that if one can recover such a key, it can break the anonymity. Thus, the security proof of anonymity is also that of the key security.

5.1 Anonymity

In Step 4 of the anonymity game (Def. 4), the adversary can select $\mathsf{m}, \mathcal{S}, \mathcal{S}'$. Due to the homomorphic property of f_{prm}, the adversary can remove m from the first coordinate, which is $\mathsf{m} \otimes f_{\mathsf{prm}}(\boldsymbol{x}^T)$ or $\mathsf{m} \otimes f_{\mathsf{prm}}((\boldsymbol{x}')^T)$, of a returned command ciphertext.

Therefore, in the context of our template construction, breaking anonymity is the same as distinguishing tuples

$$\mathsf{cmd}_S = (f_{\mathsf{prm}}(\boldsymbol{x}^T), f_{\mathsf{prm}}(\boldsymbol{ct}_1^T), \ldots, f_{\mathsf{prm}}(\boldsymbol{ct}_M^T)) \text{ and}$$
$$\mathsf{cmd}_{S'} = (f_{\mathsf{prm}}((\boldsymbol{x}')^T), f_{\mathsf{prm}}((\boldsymbol{ct}_1')^T), \ldots, f_{\mathsf{prm}}((\boldsymbol{ct}_M')^T))$$

under in a situation in which the adversary knows that there is an index $\mathsf{id} \in \mathcal{S} \setminus \mathcal{S}'$ such that $\mathsf{dk}_{\mathsf{id}}$ can recover $f_{\mathsf{prm}}(\boldsymbol{x}^T + \boldsymbol{e}_i^T)$ via the relation (3).

We transform the above problem into a distinguishing problem between legitimate and random sequences.

Definition 7. *($(f_{\mathsf{prm}}, \chi, M)$-linear distinguishing problem) For a function $f_{\mathsf{prm}}(\cdot)$ used in template construction, consider the computational problem of distinguishing the sequence*

$$(f_{\mathsf{prm}}(\boldsymbol{x}^T), f_{\mathsf{prm}}(\boldsymbol{c}_1^T), \ldots, f_{\mathsf{prm}}(\boldsymbol{c}_M^T)) \text{ and } (f_{\mathsf{prm}}(\boldsymbol{r}^T), f_{\mathsf{prm}}(\boldsymbol{c}_1^T), \ldots, f_{\mathsf{prm}}(\boldsymbol{c}_M^T))$$

where $\boldsymbol{c}_1^T, \ldots, \boldsymbol{c}_M^T$ are randomly drawn from the domain of f_{prm}.

In the former case, $\boldsymbol{x}^T$ is computed $(\boldsymbol{c}_1^T || \boldsymbol{c}_2^T || \cdots || \boldsymbol{c}_M^T)\boldsymbol{d}^T + \boldsymbol{e}^T = \boldsymbol{c}_1^T d_1 + \cdots + \boldsymbol{c}_M^T d_M + \boldsymbol{e}^T$ by a fixed secret vector $\boldsymbol{d}^T = (d_1, \ldots, d_M)^T$ and a small random error $\boldsymbol{e}^T$ from χ^r. In the latter case, $\boldsymbol{r}^T$ is random.

Theorem 1. *Using an adversary $\mathcal{A}$ that can win the anonymity game (Def. 4) with f_{prm}, noise distribution χ and dimension $2M$, above distinguishing problem with parameters $(f_{\mathsf{prm}}, \chi, M)$ can be solved with high probability.*

Proof. Fix the parameters f_{prm}, χ and M. Assume the existence of an adversary $\mathcal{A}$. Setup the anonymity game with $2M$ dimensions. The challenger then generates a $(2M) \times (2M)$ random invertible matrix U.

In the collusion query phase, suppose that the adversary requires t verification keys; we can name them $\mathsf{dk}_1, \ldots, \mathsf{dk}_t$ without loss of generality. Upon the queries, generate random M-dimensional vectors $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_t$ and set the fake verification keys to the adversary by $\mathsf{dk}_i^T = [(\boldsymbol{r}_i || \boldsymbol{u}_i)U]^T \in V^{2M}, i = 1, \ldots, t$.

Then, using the virtual secret vector $\boldsymbol{d}$ of the linear distinguishing problem, define tentative decryption keys $\mathsf{dk}_i^T = [(\boldsymbol{d} || \boldsymbol{u}_i)U]^T$ for $i = t{+}1, \ldots, M$, that are unknown to both the challenger and adversary. Upon requests from the adversary, the challenger sends the corrupted keys $\mathsf{dk}_1^T, \ldots, \mathsf{dk}_t^T$.

In the authentication query phase, the challenger generates the command ciphertext of a query $(\mathsf{m}, \mathcal{S})$ as follows. Call the problem oracle and obtain an instance $(f_{\mathsf{prm}}(\boldsymbol{y}^T), f_{\mathsf{prm}}(\boldsymbol{c}_1^T), \ldots, f_{\mathsf{prm}}(\boldsymbol{c}_M^T))$ where $\boldsymbol{y}^T$ is legitimate $\boldsymbol{x}^T$ or random $\boldsymbol{r}^T$. Denote $C = [\boldsymbol{c}_1^T || \cdots || \boldsymbol{c}_M^T]$. $F_i = f_{\mathsf{prm}}(\boldsymbol{c}_i^T)$ for $i = 1, \ldots, M$. Hence, for $i = M + 1, \ldots, M + t$, compute

$$f_{\mathsf{prm}}(C\boldsymbol{r}_i^T) = F_1 \circ r_{i,1} \otimes \cdots \otimes F_M \circ r_{i,M}$$

and

$$F_{M+i} := \begin{cases} f_{\mathsf{prm}}(C\boldsymbol{r}_i^T)^{-1} \otimes f_{\mathsf{prm}}(\boldsymbol{y}^T) \otimes f_{\mathsf{prm}}(\boldsymbol{\eta}_i^T) & (i \in S) \\ f_{\mathsf{prm}}(rand) & (i \notin S) \end{cases}$$

where *rand* means a result of random sampling from the domain of f_{prm} and $\boldsymbol{\eta}_i$ is a random noises sampled from χ^r.

For $i = M + t + 1, \ldots, 2M$, compute

$$F_{M+i} := \begin{cases} f_{\mathsf{prm}}(\boldsymbol{y}^T)^{-1} & (i \in S) \\ f_{\mathsf{prm}}(rand) & (i \notin S) \end{cases}$$

and let $(V_1, \ldots, V_{2M}) := (F_1, \ldots, F_{2M})U^{-1}$. Vector-matrix multiplications are performed using the operations $(\circ, \otimes)$, i.e., $V_j = F_1 \circ u_{1,j} \otimes \cdots \otimes F_{2M} \circ u_{2M,j}$ where $u_{i,j}$ is the (i,j)-element of U^{-1}. The command to the adversary is $\mathsf{cmd}_S = (\mathsf{m} \circ f_{\mathsf{prm}}(\boldsymbol{y}^T), V_1, \ldots, V_{2M})$.

Thus,

$$\mathsf{Vrfy}(\mathsf{vk}_i, \mathsf{cmd}_S) = \mathsf{m} \otimes f_{\mathsf{prm}}(\boldsymbol{y}^T) \otimes f_{\mathsf{prm}}^{-1}(\boldsymbol{y}^T) = \mathsf{m} \otimes \begin{cases} f_{\mathsf{prm}}(\boldsymbol{\eta}_i^T) & i = 1, \ldots, t \\ f_{\mathsf{prm}}(\boldsymbol{e}^T) & i = t+1, \ldots, M \end{cases}$$

holds for $i \in \mathcal{S}$ if the problem instance is legitimate. However, if the problem instance is random, the relations on $i = t+1, \ldots, M$ do not hold.

In the challenge query phase, for $(\mathsf{m}, \mathcal{S}_0, \mathcal{S}_1)$, the challenger returns cmd_{S_b} for $b = 0$ or 1 in the same manner and checks the adversary's response. By checking the adversary's advantage, the challenger distinguishes the problem instance. □

Instantiations of Concrete Problems: We provide instantiations of our problem (Definition 7) in DLP and LWE situations.

In the DLP situation, we set $r = 1$ and $f_{\mathsf{prm}}(x) = g^x$ and the noise variable is always zero. Thus, the problem is to distinguish

$$\{(g^x, g^{c_1}, \ldots, g^{c_M})\} \text{ and } \{(g^r, g^{c_1}, \ldots, g^{c_M})\}$$

where $(d_1, \ldots, d_M)$, $(c_1, \ldots, c_M)$, and r denote a fixed secret vector, uniformly random vector, and a random number, respectively In the legitimate situation, $x = c_1 d_1 + \cdots + c_M d_M$ holds.

To the best of our knowledge, research on this problem is scarce. However, this can be captured as a discrete logarithm analogue of the standard LWE problem. We provide the reduction to thh DDH problem.

Proposition 1. *The above distinguishing problem can be reduced to DDH.*

Proof. First, we prove that an algorithm $\mathcal{A}_M$ to distinguish the above $M+1$ dimensional vectors can solve the $M = 1$ problem. We can construct an algorithm $\mathcal{A}_1$ to solve the $M = 1$ problem. Before calling the oracle, $\mathcal{A}_1$ samples a virtual secrets $(d_2, \ldots, d_M)$ and fixes them. For a 1-instance (g^x, g^{c_1}), construct M-instance by $(g^y, g^{c_1}, \ldots, g^{c_M})$ with $g^y = g^x \cdot g^{c_2 d_2 + \cdots + c_M d_M}$ with randomly generated $c_2, \ldots, c_M$. If g^x is a legitimate $g^{c_1 d_1}$ (resp. random g^r), g^y is a legitimate (resp. random) number.

Next, we prove that $\mathcal{A}_1$ can distinguish DH instances. Fix a generator g and let (g^a, g^b, g^c) be a sample where $c = ab$ or random. Note that for any randomly generated d_1, d_2, the relation $(g^{d_1} \cdot (g^b)^{d_2})^a = ((g^a)^{d_1} \cdot (g^c)^{d_2})$ holds if $c = ab$. Thus, a sequence of 1-instances with secret a can be generated by $(g^x, g^{c_1}) = (g^{d_1} \cdot (g^b)^{d_2}, (g^a)^{d_1} \cdot (g^c)^{d_2})$. If $c = ab$, it is a legitimate sample. On the other hand, if $c \neq ab$, the second component is $g^{c_1} = g^{ad_1 + abd_2} \cdot g^{(c-ab)d_2}$. Thus, the randomness of d_2 makes its distribution random. □

In the lattice setting with $\mathsf{prm} = \boldsymbol{p} \in \mathbb{Z}_q^r$ and $f_{\boldsymbol{p}}(\boldsymbol{x}^T) = \boldsymbol{p}\boldsymbol{x}^T$, the problem is to distinguish

$$(\boldsymbol{p}\boldsymbol{x}^T, \boldsymbol{p}\boldsymbol{c}_1^T, \ldots, \boldsymbol{p}\boldsymbol{c}_M^T) \text{ and } (\boldsymbol{p}\boldsymbol{r}^T, \boldsymbol{p}\boldsymbol{c}_1^T, \ldots, \boldsymbol{p}\boldsymbol{c}_M^T)$$

where $\boldsymbol{x}^T$ is computed by $\sum_{i=1}^M d_i \boldsymbol{c}_i^T + \boldsymbol{e}^T$ by a secret vector $\boldsymbol{d} = (d_1, \ldots, d_M)$ and an error vector $\boldsymbol{e}^T$, and $\boldsymbol{r}^T$ is a random vector. This is a sample $(\boldsymbol{a}, \boldsymbol{a}\boldsymbol{s}^T + \eta)$ of the decision LWE problem where $\boldsymbol{a} = (\boldsymbol{p}\boldsymbol{c}_1^T, \ldots, \boldsymbol{p}\boldsymbol{c}_M^T)$, $\boldsymbol{s} = (d_1, \ldots, d_M)$ and $\eta = \boldsymbol{p}\boldsymbol{e}^T$. The distribution of η is a scaled Gaussian if each coordinate of $\boldsymbol{e}$ is drawn from a Gaussian.

5.2 Unforgeability

We present a t-unforgeable ABA scheme using a transformation by adding signatures. Notably, the template construction does not have 1-unforgeability because of its homomorphic property. In fact, for

$$\mathsf{cmd}_{\mathcal{S}} = (\mathsf{m} \otimes f_{\mathsf{prm}}(\boldsymbol{x}^T), f_{\mathsf{prm}}(\mathsf{ct}_1^T), \ldots, f_{\mathsf{prm}}(\mathsf{ct}_M^T))$$

that targets $\mathcal{S}$ selected by an adversary having only $\mathsf{vk}_{\mathsf{id}}$, $\mathsf{cmd}_{\mathcal{S}} = (\mathsf{m} \otimes f_{\mathsf{prm}}(\boldsymbol{x}^T) \otimes \mathsf{c}, f_{\mathsf{prm}}(\mathsf{ct}_1^T), \ldots, f_{\mathsf{prm}}(\mathsf{ct}_M^T))$ is a legitimate ciphertext of the shifted message $\mathsf{m} \otimes \mathsf{c}$ accepted by some $\mathsf{id}' \in S \setminus \{\mathsf{id}\}$.

A simple transformation technique from a CPA-secure public-key BE to a CCA1-secure one is known [22]. Following their notions and techniques, we propose a method of transformation from our template ABA to an unforgeable ABA.

Definition 8. *(Transformation) For an ABA scheme* $ABA = (\mathsf{Setup}, \mathsf{Join}, \mathsf{Auth}, \mathsf{Vrfy})$ *and a strongly and existentially unforgeable signature* $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$*, the transformation of ABA, which we denote* ABA_Σ *is defined as follows.*

- $ABA_\Sigma.\mathsf{Setup}(1^\lambda, N, \mathcal{D}) \to (\mathsf{ak}, \mathsf{pk}, \mathsf{sk})$: *Run* $ABA.\mathsf{Setup}(1^\lambda, N, \mathcal{D}) \to \mathsf{ak}$ *and* $\Sigma.\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$.
- $ABA_\Sigma.\mathsf{Join}(\mathsf{ak}, \mathsf{id}) \to \mathsf{vk}_{\mathsf{id}}$: *Run* $ABA.\mathsf{Join}(\mathsf{ak}, \mathsf{id})$ *and let* $\mathsf{vk}_{\mathsf{id}} = (ABA.\mathsf{vk}_{\mathsf{id}}, \mathsf{pk})$.
- $ABA_\Sigma.\mathsf{Auth}(\mathsf{ak}, \mathsf{m}, \mathcal{S}) \to (\mathsf{cmd}_{\mathcal{S}}, \sigma)$: *Execute* $ABA.\mathsf{Auth}(\mathsf{ak}, \mathsf{m}, \mathcal{S}) \to \mathsf{cmd}_{\mathcal{S}}$. *Generate the signature for the base command* $\Sigma.\mathsf{Sign}(\mathsf{sk}, \mathsf{cmd}_{\mathcal{S}}) \to \sigma$.
- $ABA_\Sigma.\mathsf{Vrfy}(\mathsf{vk}_i, (\mathsf{cmd}_{\mathcal{S}}, \sigma)) \to \mathsf{m}/\mathsf{reject}$: *Check the signature* $\Sigma.\mathsf{Vrfy}(\mathsf{pk}, \sigma, \mathsf{cmd}_{\mathcal{S}})$. *If the check fails, return* reject. *Passing the verifications, execute* $ABA.\mathsf{Vrfy}(\mathsf{vk}_{\mathsf{id}}, \mathsf{cmd}_S)$ *and return the result.*

The security proof of unforgeability is straightforward. In the security game (Definition 3), an adversary can get verification keys and $\{(\mathsf{cmd}_{\mathcal{S}}, \sigma)\}$ upon one's queries. Suppose one can forge a command pair $(\mathsf{cmd}'_{\mathcal{S}'}, \sigma')$ with $(\mathsf{m}', \mathsf{id}')$ such that $\Sigma.\mathsf{Vrfy}(\mathsf{pk}, \sigma', \mathsf{cmd}'_{S'})$ returns accept and $ABA.\mathsf{Vrfy}(\mathsf{vk}_{\mathsf{id}'}, \mathsf{cmd}'_{S'})$ returns m'.

Forging is split into two cases. If $\mathsf{cmd}'_{\mathcal{S}'}$ is not equal to any commands returned from the challenger in the authentication query phase $(\mathsf{cmd}'_{\mathcal{S}'}, \sigma')$ is a valid pair to break the strong unforgeability of the signature game, which is assumed to be hard.

Next, consider a situation where $\mathsf{cmd}'_{\mathcal{S}'}$ is equal to $\mathsf{cmd}_{\mathcal{S}_a}$, one of the returned commands in the authentication queries. We show that this situation is impossible. Recall that the corresponding message m' and m_a in the commands cannot be equal by the requirement $\mathsf{m} \notin M_v$ in Step 3. Thus, the first element of $\mathsf{cmd}'_{S'} = \mathsf{cmd}_{\mathcal{S}_a}$ is $\mathsf{m}' \otimes f_{\mathsf{prm}}(\boldsymbol{x}'^T) = \mathsf{m}_a \otimes f_{\mathsf{prm}}(\boldsymbol{x}_a^T)$, which are different representations of different messages. Thus, the verification results by $\mathsf{vk}_{\mathsf{id}'}$ must satisfy $ABA.\mathsf{Vrfy}(\mathsf{vk}_{\mathsf{id}'}, \mathsf{cmd}'_{S'}) = \mathsf{m}'$ and $ABA.\mathsf{Vrfy}(\mathsf{vk}_{\mathsf{id}'}, \mathsf{cmd}_{S_a}) = \mathsf{m}_a$. This contradicts the requirement $\mathsf{m}' \neq \mathsf{m}_a$ and $\mathsf{cmd}'_{\mathcal{S}'} = \mathsf{cmd}_{\mathcal{S}_a}$.

Therefore, forging a command ciphertext is hard because of the strong unforgeability of the signature. Clearly this construction inherits the other security properties of the base ABA, i.e., anonymity and message security.

5.3 Message Security

This section discusses the hardness of distinguishing messages from command ciphertexts without the keys of target devices.

We introduce a game-based definition of command indistinguishability.

Definition 9. *(IND-CPA security of ABA with t-collusion) Consider the following game between an adversary* A *and a challenger* C.

0: Share N *the number of participant devices and* pp *the public parameter.* C *runs* $\mathsf{Setup}(1^\lambda, N, \mathcal{D}) \to \mathsf{ak} = (\mathsf{ek}, \{\mathsf{dk}_i\})$.

1: (Collusion query) A *selects* $\mathsf{id} \in [N]$ *and sends it to the challenger.* C *runs* $\mathsf{Join}(\mathsf{ak}, \mathsf{id}) \to \mathsf{vk}_{\mathsf{id}}$, *adds* id *to* W *and sends* $\mathsf{vk}_{\mathsf{id}}$ *to* A. A *can repeat this step until the number of colluded devices is less than* t.

2: (Challenge ciphertext) A *selects a set of target devices* $\mathcal{S} \subset [N], \mathcal{S} \cap W = \phi$ *and two messages* $\mathsf{m}_0, \mathsf{m}_1$ *and send them to* C. C *generates a ciphertext* $\mathsf{cmd}_{\mathcal{S}}$ *from* m_b *where* b *is randomly chosen from* $\{0, 1\}$ *and return the ciphertext to* A.

3: (Encryption query) A *can ask* C *to encrypt* $(\mathsf{m}', \mathcal{S}')$ *and receive the ciphertext* $\mathsf{cmd}_{\mathcal{S}'}$

4: A *guesses* b' *from* cmd_S.

The advantage of A *is defined by* $2 \cdot |\Pr[b = b'] - 1/2|$. *The scheme is said to be IND-CPA secure if the advantage is negligible.*

In our template construction, the challenge ciphertext is given by the following form

$$\mathsf{cmd}_{\mathcal{S}} = (\mathsf{m}_b \otimes f_{\mathsf{prm}}(\boldsymbol{x}^{\boldsymbol{T}}), f_{\mathsf{prm}}(\boldsymbol{c}_1^T), \ldots, f_{\mathsf{prm}}(\boldsymbol{c}_M^T))$$

and a command ciphertext with known m' in the encryption query phase is

$$\mathsf{cmd}_{\mathcal{S}'} = (\mathsf{m}' \otimes f_{\mathsf{prm}}(\boldsymbol{x}^T), f_{\mathsf{prm}}((\boldsymbol{c}_1')^T), \ldots, f_{\mathsf{prm}}((\boldsymbol{c}_M')^T)).$$

We present hardness results to distinguish the messages in DLP and LWE situations. The proofs are separated into each instantiations via the use of error vectors though proof outlines are similar.

Theorem 2. *The IND-CPA security of ABA in the DLP instantiation can be reduced to the DDH assumption.*

Proof. Suppose $\mathcal{A}$ can distinguish the command ciphertext of ABA with M dimensional vectors, N devices, $f_g(a) = g^a$ and t colluded devices. We show how the challenger distinguishes an instance of the DDH assumption (g, g^r, g^x, g^c) where $c = rx$ or a random number.

Let $\overline{\mathsf{dk}}_i^T, i \in [N]$ be fake decryption keys of the base MRE that the challenger randomly generates. Upon the adversary's collusion queries, the challenger returns a sequence of fake keys. Upon the adversary's set $\mathcal{S}$, a random number $\overline{x}$ and a row vector CT are generated such that

$$CT \cdot \overline{\mathsf{dk}}_i^T = \begin{cases} \overline{x} & (i \in \mathcal{S}) \\ rand & (i \notin \mathcal{S}) \end{cases}$$

where $rand$ is a random number except for $\overline{x}$. Then, parse $CT = (\mathsf{ct}_1, \ldots, \mathsf{ct}_M)$.

Upon the messages $\mathsf{m}_0, \mathsf{m}_1$, randomly choose m_b and let the returning command be

$$\mathsf{cmd}_{\mathcal{S}'} = (\mathsf{m}_b \cdot g^c, g^{r\mathsf{ct}_1}, \ldots, g^{r\mathsf{ct}_M}).$$

Let dk_i^Ts be the true keys assumed to be used in a virtual encryption system from the adversary's view. They satisfy $CT \cdot \mathsf{dk}_i^T = x$. The relation between the fake and true keys is $\overline{\mathsf{dk}}_i^T = \mathsf{dk}_i^T \cdot (\overline{x}/x)$.

Upon the adversary's encryption query $(\mathsf{m}', \mathcal{S}')$, the challenger again generate a random number $\overline{x'}$ and a matrix CT' such that

$$CT' \cdot \overline{\mathsf{dk}_i}^T = \begin{cases} \overline{x'} & (i \in \mathcal{S}) \\ rand & (i \notin \mathcal{S}) \end{cases}$$

and the command ciphertext

$$\mathsf{cmd}_{\mathcal{S}'} = (\mathsf{m}' \cdot g^{x \cdot (\overline{x'}/\overline{x})}, g^{\mathsf{ct}'_1}, \ldots, g^{\mathsf{ct}'_M}).$$

Since we have g^x, $\overline{x}$ and $\overline{x'}$, the first element is easily computable. For true key $\mathsf{dk}_i^T = (d_{i,1}, \ldots, d_{i,M})^T$, we have

$$\prod_{i=1}^{M} (g^{\mathsf{ct}'_i})^{d_{i,j}} = g^{CT' \cdot \mathsf{dk}_i} = g^{\overline{\mathsf{dk}_i}^T \cdot (x/\overline{x})} = g^{\overline{x'} \cdot (x/\overline{x})}$$

Thus, this is a legitimate command that encrypts m'.

Finally, the adversary guesses b' and the advantage $|\Pr[b = b'] - 1/2|$ should be high if $c = rx$ and small if c is random. □

Besides the DLP situation, we provide the security of our lattice-based ABA to the hardness of the decision LWE in the lattice case.

Theorem 3. *Let (n, q, m) be the LWE parameter and the noise distribution be χ. The hardness of the decision LWE guarantees the IND-CPA security of ABA with lattice instantiation with noise distribution $\chi + \chi$ and vector dimension $M = 2n$. In addition, assume the domain of $f_{\mathsf{prm}}(\cdot)$ has dimension $r = m$.*

Proof. Let $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m$ and $\boldsymbol{b}^T = (b_1, \ldots, b_m)^T$ be sample instances of the decision LWE. $b_i = \boldsymbol{a}_i \boldsymbol{s}^T + e_i \bmod q$ or a random number in $\mathbb{Z}_q$. We also let $\boldsymbol{e}^T = (e_1, \ldots, e_m)$ from the noise distribution χ^m.

Let $\mathcal{A}$ be an adversary that can distinguish two messages of ABA working with $2n$-dimensional vectors, N devices, and t colluded devices. The template function $f_{\boldsymbol{p}}(\boldsymbol{x}^T) = \boldsymbol{p}\boldsymbol{x}^T$ is defined over m-dimensional vectors, where $\boldsymbol{p}$ denotes a small vector to be kept secret but random for security.

For collusion queries, we can assume that the IDs are $\mathsf{id} = 1, \ldots, t$ without loss of generality and let fake verification keys be $2n$-dimensional vectors $\overline{\mathsf{vk}_i}^T = [(\boldsymbol{r}_i || \boldsymbol{u}_i) U]^T$, where $\boldsymbol{r}_i$s and U are random vectors from $\mathbb{Z}_q^n$ and a random matrix in $\mathbb{Z}_q^{2n \times 2n}$, respectively. In addition, for other IDs, we tentatively set $\overline{\mathsf{vk}_i}^T = [(\boldsymbol{s} || \boldsymbol{u}_i) U]^T$, which is unknown to both the challenger and adversary because $\boldsymbol{s}$ is the secret vector of the LWE instance.

Upon the adversary's request $\mathsf{m}_0, \mathsf{m}_1$ and $\mathcal{S}$ in Step 2, the challenge ciphertext is constructed using LWE samples $\boldsymbol{a}_i, \boldsymbol{b}$ and randomly generated m_b as follows. For the row vectors $\boldsymbol{a}_i$, construct the $r \times n$ matrix and parse it into n column vectors

$$\begin{bmatrix} \boldsymbol{a}_1 \\ \vdots \\ \boldsymbol{a}_r \end{bmatrix} = \left[A_1^T | \cdots | A_n^T \right].$$

The command is

$$\mathsf{cmd}_{\mathcal{S}} = (\mathsf{m}_b + \boldsymbol{b}\boldsymbol{p}^T, [A_1 \boldsymbol{p}^T, \ldots, A_n \boldsymbol{p}^T, F_{n+1}, \ldots, F_{2n}] U^{-1} \bmod q).$$

The latter elements are defined by

$$F_{n+i} = \begin{cases} rand & (i \notin \mathcal{S}) \\ 0 & (i \in \mathcal{S} \cap \{t+1, \ldots, N\}) \text{ //non-colluded keys} \\ \boldsymbol{b}\boldsymbol{p}^T - \sum_{i=1}^{n} r_i A_i \boldsymbol{p}^T + \boldsymbol{\eta_i}\boldsymbol{p}^T & (i \in \mathcal{S} \cap [t]) \text{ //colluded keys} \end{cases}$$

where $\boldsymbol{\eta}_i$ is a vector from $D^r_{\mathbb{Z},\sigma}$. For this setting, the results of the target and nontarget device verifications are as follows. For $i \in S \cap [t]$, we have

$$\begin{aligned} &(A_1\boldsymbol{p}^T, \ldots, A_n\boldsymbol{p}^T, F_{n+1}, \ldots, F_{2n})U^{-1} \cdot (\overline{\mathsf{vk}_i}^T) \\ &= (A_1\boldsymbol{p}^T, \ldots, A_n\boldsymbol{p}^T, F_{n+1}, \ldots, F_{2n}) \cdot (\boldsymbol{r}_i||\boldsymbol{u}_i)^T = \boldsymbol{b}\boldsymbol{p}^T + \boldsymbol{\eta}_i\boldsymbol{p}^T. \end{aligned}$$

Thus,

$$\mathsf{Vrfy}(\overline{\mathsf{vk}}_i, \mathsf{cmd}_{\mathcal{S}}) = \mathsf{m}_b - \boldsymbol{\eta}_i\boldsymbol{p}^T \tag{5}$$

For $i \in S \cap ([N] \setminus [t])$, we have

$$\begin{aligned} &(A_1\boldsymbol{p}^T, \ldots, A_n\boldsymbol{p}^T, F_{n+1}, \ldots, F_{2n})U^{-1} \cdot (\overline{\mathsf{vk}_i}^T) \\ &= (A_1\boldsymbol{p}^T, \ldots, A_n\boldsymbol{p}^T, F_{n+1}, \ldots, F_{2n}) \cdot (\boldsymbol{s}_i||\boldsymbol{u}_i)^T = \sum_{i=1}^{n} s_i\boldsymbol{a}_i\boldsymbol{p}^T \end{aligned}$$

Thus,

$$\mathsf{Vrfy}(\overline{\mathsf{vk}}_i, \mathsf{cmd}_{\mathcal{S}}) = \mathsf{m}_b + \boldsymbol{b}\boldsymbol{p}^T - \sum_{i=1}^{n} s_i\boldsymbol{a}_i\boldsymbol{p}^T = \mathsf{m}_b + (\boldsymbol{e} + \boldsymbol{\eta}_i)\boldsymbol{p}^T. \tag{6}$$

For encryption queries $(\mathsf{m}', \mathcal{S}')$, the command can be constructed similarly. Therefore, the LWE samples can be converted to the command. If the LWE instance is legitimate (resp. random), the advantage of $\mathcal{A}$ should be high (resp. low). Thus, $\mathcal{A}$ can solve the decision LWE problem.

Finally, it is necessary to pay attention to the noise distribution. In the above transformation, the noises (5) and (6) differ slightly. By replacing both $\boldsymbol{e}^T + \boldsymbol{\eta}_i^T$ and $\boldsymbol{\eta}_i^T$ by another noise $(\boldsymbol{\eta}')_i^T$ with larger derivations, the distinguishing hardness is amplified. If $\boldsymbol{e}^T$ and $\boldsymbol{\eta}_i^T$ are discrete Gaussians of variance σ^2, then we can take $(\boldsymbol{\eta}')_i^T$ the discrete Gaussians of variance $2\sigma^2$. The proof of the relation between noise parameters is completed. □

5.4 Key Security

We now discuss security of vefication keys. Suppose an attacker can collect verification keys $\{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id} \in W}$ from colluded devices. The goal is not only to find a verification key vk_i for some $i \notin W$ but also to generate an alternative key, i.e., i.e., a key vk'_i that $\mathsf{Vrfy}(\mathsf{vk}'_i, \mathsf{cmd}_S)$ outputs the correct value with high probability.

If one can recover such a key, anonymity can be broken as follows. In the anonymity game (Def. 4), suppose that an adversary can generate an alternative key vk'_i for some device $i \notin W$ using the colluded keys in Step 2. The adversary then randomly selects $\mathcal{S}_0, \mathcal{S}_1$ but $i \in \mathcal{S}_1$ and $i \notin \mathcal{S}_0$ hold. From the result of $\mathsf{Vrfy}(\mathsf{vk}'_i, \mathsf{cmd}_{\mathcal{S}_b})$, the adversary can distinguish b.

Therefore, the security proof of anonymity is also the proof of key security.

6 Concatenation of ABAs

Our template construction in the previous sections is under the atomic framework; thus, the ciphertext length is $\Omega(N)$. As aforementioned, this is useless for controlling millions of devices. Using the homomorphic properties of the base function f_{prm}, we can concatenate ABAs to control many devices in practice.

The sequential concatenation of small-sized ABAs is a simple technique for reducing the ciphertext length by restricting the choice of target devices. The plain construction, which is immediately found to be insecure, is described as follows.

Definition 10. *(Sequential concatenation of ABAs) Consider K ABAs. Let them be $ABA_j = (\mathsf{Setup}_j, \mathsf{Join}_j, \mathsf{Auth}_j, \mathsf{Vrfy}_j)$ and N_j be the maximum number of devices controlled by the j-th ABA. The concatenated ABA is defined as follows. Each device is indicated by a vector $\mathsf{id} = (i_1, \ldots, i_K)$.*

- *$ABA.\mathsf{Setup}$: Execute $ABA_j.\mathsf{Setup}(1^\lambda, N, \mathcal{D}) \to \mathsf{ak}_j$ for all $j \in [K]$. The order of execution does not matter and let $\mathsf{ak} := \{\mathsf{ak}_1, \ldots, \mathsf{ak}_K\}$.*
- *$ABA.\mathsf{Join}(\mathsf{ak}, \mathsf{id})$: For $\mathsf{id} = (i_1, \ldots, i_K)$, execute $ABA_j.\mathsf{Join}(\mathsf{ak}_j, i_j) \to \mathsf{vk}_{j,i_j}$. The verification key is the concatenation $\mathsf{vk}_{\mathsf{id}} = (\mathsf{vk}_{1,i_1}, \ldots, \mathsf{vk}_{K,i_K})$.*
- *$ABA.\mathsf{Auth}(\mathsf{ak}, \mathsf{m}, \mathcal{S})$: The set of target devices is indicated by $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_K$ where $\mathcal{S}_j \subset [N_j]$. The command ciphertext $\mathsf{cmd}_{\mathcal{S}}$ is the concatenation of $\mathsf{cmd}_{\mathcal{S}_j} = \mathsf{Auth}(\mathsf{ak}_j, \mathsf{m}, \mathcal{S}_j)$ for $j = 1, \ldots, K$, and it is broadcast.*
- *$ABA.\mathsf{Vrfy}(\mathsf{vk}_{\mathsf{id}}, \mathsf{cmd}_{\mathcal{S}}) \to \mathsf{m}/\mathsf{reject}$. Check whether $\mathsf{Vrfy}(\mathsf{vk}_{j,i_j}, \mathsf{cmd}_{\mathcal{S}_j})$ for all j. If all the verification has been accepted, output m, if otherwise, output reject.*

Assuming the anonymity of the base ABA_j, the concatenated ABA also has anonymity on two sets of certain limited forms. For instance, consider two sets $\mathcal{S}_1 \times \cdots \times \mathcal{S}_K$ and $\mathcal{S}'_1 \times \cdots \times \mathcal{S}'_K$ that are only their j-th coordinates differ. We can show that the two command ciphertexts that encode the sets are indistinguishable due to the anonymity of base ABA_j.

The concatenated ABA has certain insecurities. The message contents are not secure because a nontarget device can recover the message. In fact, Vrfy returns the message whenever $\mathsf{id}_j \in \mathcal{S}_j$. In addition, one can break anonymity properties in the senses of both standard notion (Definition 4) and another notion that we will name the single anonymity.

6.1 Rearranging Attack

The sequential concatenation of ABAs is easily broken by the rearrangement of the verification keys. Consider the concatenation of $K = 2$ ABAs with $N_1 = N_2 = 2$. The composed ABA can control $N_1 N_2 = 4$ devices and we name them by $\mathsf{id} = (1,1), (1,2), (2,1)$ and $(2,2)$. Suppose that $(1,1)$ and $(2,2)$ are colluded. If an attacker has $\mathsf{vk}_{1,1} = (\mathsf{vk}_{1,1}, \mathsf{vk}_{2,1})$ and $\mathsf{vk}_{2,2} = (\mathsf{vk}_{1,2}, \mathsf{vk}_{2,2})$, then it can generate other verification keys $\mathsf{vk}_{1,2} = (\mathsf{vk}_{1,1}, \mathsf{vk}_{2,2})$ and $\mathsf{vk}_{2,1} = (\mathsf{vk}_{1,2}, \mathsf{vk}_{2,1})$ via the recombination of components. Thus, the attacker can recover any legitimate ciphertext and determine which devices are in the target set. For instance, in the anonymity game, an adversary can select $\mathcal{S}_0 = \{(1,2)\}$ and $\mathcal{S}_1 = \{(2,1)\}$ that satisfies (1). Thus, $\mathsf{cmd}_{\mathcal{S}_b}$ can be easily verified to distinguish.

This attack is quite strong because it uses neither the properties of encrypting nor homomorphic functions. Below, we show that another type of anonymity can be broken even if one cannot recover other verification keys.

6.2 Single Anonymity in Restricted Device Selection

Besides the standard anonymity (Definition 4), there is a situation in which one can break another type of anonymity from only the results of verification on the colluded devices. We first provide an example.

Again consider the concatenation of $K = 2$ ABAs with $N_1 = N_2 = 2$. Denote the target indication by $(d_1, d_2) \in \{1, 2, *\}^2$ where $*$ is the wild-card; for example $(1, *)$ represents the set $\{1\} \times \{1, 2\} = \{(1, 1), (1, 2)\}$. Nine patterns are possible as the rows in the table below.

(d_1, d_2)	(1,1)	(1,2)	(2,1)	(2,2)
(1,1)	•			
(1,2)		•		
(2,1)			•	
(2,2)				•

(d_1, d_2)	(1,1)	(1,2)	(2,1)	(2,2)
(1,*)	•	•		
(2,*)			•	•
(*,1)	•		•	
(*,2)		•		•
(*,*)	•	•	•	•

Assume $\mathsf{id} = (1, 2)$ and $(2, 1)$ are colluded and that the attacker can obtain $\mathsf{vk}_{1,2}$ and $\mathsf{vk}_{2,1}$. If the attacker decrypts a command ciphertext and knows that it targets both $\mathsf{id} = (1, 2)$ and $(2, 1)$. Then, the selection $(d_1, d_2) = (*, *)$ is revealed from the table, and the attacker can know $\mathsf{id} = (1, 1)$ and $(2, 2)$ are also target devices. We emphasize that such an attack is possible even if the other verification keys cannot be recovered. This is independent of the cryptographic security of the primitives.

The above example illustrates how we consider anonymity in a situation in which the freedom of target device selection is limited.

In the notion of standard anonymity (Definition 4), a situation in which no information is leaked except for the corrupted devices is considered via the indistinguishability property. We believe that the validity of this definition is justified by an implicit assumption that the target set is uniformly chosen from $2^{[N]}$ and information leakage is always caused by cryptographic vulnerabilities.

In our limited-freedom situation, information can be leaked regardless of the cryptographic vulnerability. We introduce the notion of single anonymity to formulate this type of anonymity.

Definition 11. *An ABA is said to have single anonymity (SA) if for any $W \subsetneqq [N]$ a pattern of $\{\mathsf{Vrfy}(\mathsf{vk}_{\mathsf{id}}, \mathsf{cmd}_S)\}_{\mathsf{id} \in W}$ does not reveal the information whether $\mathsf{id}' \overset{?}{\in} \mathcal{S}$ for some $\mathsf{id}' \notin W$.*

By the notation $\mathsf{id}' \overset{?}{\in} \mathcal{S}$, we mean information to determine $\mathsf{id}' \in \mathcal{S}$ or $\mathsf{id} \in \mathcal{S}$ with 100% certainty. A relaxed version of this definition, e.g., based on probabilistic bias, can be considered because this condition is quite tight; by the next proposition, any subset of $[N]$ must be selectable. One of our future works is to investigate the relationship between the restriction of target device selection (that bounds ciphertext length) and the strength of anonymity.

Proposition 2. *If the freedom to select target devices is limited, it does not have SA.*

Proof. Let $\mathcal{D} \subset 2^{[N]}$ be a family of sets that can be specified as targets. We say $\mathcal{S} \subset [N]$ is selectable (resp. unselectable) if $\mathcal{S} \in \mathcal{D}$ (resp. $\mathcal{S} \notin \mathcal{D}$.)

We separate the situation. First, assume that there are nselectable $\mathcal{S}$ and selectable $\mathcal{S}'$ such that $\mathcal{S}' \supset \mathcal{S}$. Considering the minimum such $\mathcal{S}'$, i.e., any proper subset of $\mathcal{S}'$ is unselectable. Suppose all devices in $\mathcal{S} \cup ([N] \setminus \mathcal{S}')$ have been colluded and the server broadcasts a command $\mathsf{cmd}_{S'}$ that

targets all devices in $\mathcal{S}'$. The adversary can verify the commands in the infected devices and know if they are in target (resp. non-target.) for $\mathsf{id} \in \mathcal{S}$ (resp. $\mathsf{id} \in [N] \setminus \mathcal{S}'$).

With the restriction of the choice of $\mathcal{S}$, the adversary also determines that the devices in $\mathcal{S}' \setminus \mathcal{S}$ are in the target without using any cryptographic attacks.

Consider another situation in which for any selectable set $\mathcal{S}$, its subset is selectable. This situation implies that $[N]$ is divided into $\mathcal{T}_1 \cup \cdots \cup \mathcal{T}_k$ such that $\mathcal{T}_i \cap \mathcal{T}_j = \phi$ and any subsets of $\mathcal{T}_i$ are selectable. Thus, if one can know a device $\mathsf{id} \in \mathcal{T}_i$ is a target, all the devices in $\mathcal{T}_j, (j \neq i)$ that are not selectable can be deduced. □

Of note, neither t-anonymity nor SA implies each other. In addition, even if we construct a t-anonymity ABA for a certain reasonably high t, information leakage in the sense of SA from a pattern is possible. This is a motivation for our construction strategy in the next section using a simple known technique and algebraic analysis.

6.3 Modification Against Recombination Attack

The simple concatenated construction has certain insecurities. The anonymity and unforgeability of the concatenated ABAs are broken by rearranging the colluded keys and a nontarget device can recover the message. Based on the abovementioned discussion, retaining complete anonymity in the sense of both standard anonymity and single anonymity is impossible because the freedom of target device selection is limited.

We employ two modifications to strengthen the security. The first idea is based on the well-known secret sharing scheme to distribute m into K shares and recover them in target devices via the homomorphic property of f_{prm}. The other idea is based on Agrawal et al.'s [1] inner product encryption. We use this technique as a mechanism to increase the amount of information required to break anonymity by adding variables. We remark that this does not provide complete anonymity but increases the practical security.

Below, we provide a template construction modified from Definition 10.

Definition 12. *(A modified concatenated template construction)*
• $\mathsf{Setup}(1^\lambda, K, \mathcal{D}) \to \mathsf{ak}$: *Fix a prime field* $\mathbb{Z}_q$ *and a dimension* M *of base MREs. Execute* $\mathsf{MRE}_j.\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{ek}_j, \{\mathsf{dk}_{j,i}^T\}_{i\in[N_j]})$ *for* $j \in [K]$*, where* $\mathsf{ek}_j := \{\mathsf{dk}_{j,i}^T\}$*. Generate random matrices* $A_{j,i} \xleftarrow{\$} \mathbb{Z}_q^{r\times M}$ *for* $j \in [K], i \in [N_j]$*, a random invertible matrix* $W \in \mathbb{Z}_q^{2M\times 2M}$ *and a random vector* $\boldsymbol{u}^T \in \mathbb{Z}_q^r$*. The key is* $\mathsf{ak} = (\{\mathsf{dk}_{j,i}^T\}, \{\mathsf{ek}_j\}, \{A_{j,i}\}, W, \boldsymbol{u}^T)$.
• $\mathsf{Join}(\mathsf{ak}, \mathsf{id}) \to \mathsf{vk}_{\mathsf{id}}$: *For a device* $\mathsf{id} = (i_1, i_2, \ldots, i_K)$*, generate a random vector* $\mathsf{uk}_{\mathsf{id}}$ *such that* $\sum_{j=1}^K A_{j,i_j} \mathsf{uk}_{\mathsf{id}}^T = \boldsymbol{u}^T \pmod q$*. The verification key is* $\mathsf{vk}_{\mathsf{id}} = (W(\mathsf{dk}_{1,i_1}||\mathsf{uk}_{\mathsf{id}})^T, \ldots, W(\mathsf{dk}_{K,i_K}||\mathsf{uk}_{\mathsf{id}})^T)$.
• $\mathsf{Auth}(\mathsf{ak}, \mathsf{m}, \mathcal{S}) \to \mathsf{cmd}_{\mathcal{S}}$: *Suppose the target devices are indicated by* $\mathcal{S}_1 \times \cdots \times \mathcal{S}_K \subset \prod[N_j]$*. Pick random vectors* $\boldsymbol{t}_j^T \in \mathbb{Z}_q^r$ *and let* $\boldsymbol{x}^T := \boldsymbol{t}_1^T + \cdots + \boldsymbol{t}_K^T$*. Generate random matrices* $CT_{j,i}$ *(*$j \in [M], i \in [N_j]$*) such that*

$$CT_{j,i} \cdot \mathsf{dk}_{j,\ell} = \begin{cases} \boldsymbol{t}_j^T + \boldsymbol{e}_{j,i}^T \ (i = \ell \text{ and } i \in S_j) \\ rand \qquad\quad (i \notin S_j) \end{cases}$$

where rand represents a random element far from $\boldsymbol{t}_j^T$*. Define the matrix* $C_{j,i} := (CT_{j,i}||A_{j,i})W^{-1}$ *and split it into the* $2M$ *column vectors by* $C_{j,i} = (\boldsymbol{c}_{j,i,1}^T||\cdots||\boldsymbol{c}_{j,i,2M}^T)$*. The command ciphertext* $\mathsf{cmd}_{\mathcal{S}}$ *is* $\mathsf{m} \otimes f_{\mathsf{prm}}(\boldsymbol{x}^T + \boldsymbol{u}^T)$ *and the sequence* $\{f_{\mathsf{prm}}(\boldsymbol{c}_{j,i,\ell}^T)\}$.

• $\mathsf{Vrfy}(\mathsf{vk}_{\mathsf{id}}, \mathsf{cmd}_{\mathcal{S}})$: *For* $\mathsf{id} = (i_1, \ldots, i_K)$, *denote* $\mathsf{vk}_{\mathsf{id}} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_K)$ *and let* ℓ*-th element of* $\boldsymbol{v}_j$ *be* $v_{j,\ell}$. *For the command* $(\mathsf{m} \otimes f_{\mathsf{prm}}(\boldsymbol{x}^T + \boldsymbol{u}^T), \{F_{j,i,\ell}\})$, *compute* $T_j = \sum_{\ell=1}^{2M} v_{j,\ell} \otimes F_{j,i_j,\ell}$ *and* $\mathsf{m} \otimes f_{\mathsf{prm}}(\boldsymbol{x}^T + \boldsymbol{u}^T) \otimes (T_1 \otimes \cdots \otimes T_K)^{-1}$.

The correctness is straightforward. For a target $\mathsf{id} = (i_1, \ldots, i_K)$, each T_j and the sum in the sense of $\otimes$ are

$$
\begin{aligned}
&T_j = \sum_{\ell=1}^{2M} v_{j,\ell} \otimes f_{\mathsf{prm}}(\boldsymbol{c}_{j,i_j,\ell}^T) = f_{\mathsf{prm}}\left(\sum_{\ell=1}^{2M} \boldsymbol{c}_{j,i_j,\ell}^T v_{j,\ell}\right) \\
&= f_{\mathsf{prm}}((CT_{j,i_j}||A_{j,i_j})W^{-1} \cdot W(\mathsf{dk}_{j,i_j}||\mathsf{uk}_{\mathsf{id}})) = f_{\mathsf{prm}}(\boldsymbol{t}_j^T + \boldsymbol{e}_{j,i}^T + A_{j,i_j}\mathsf{uk}_{\mathsf{id}}), \\
&\text{and } \; T_1 \otimes \cdots \otimes T_K = f_{\mathsf{prm}}\left(\textstyle\sum_{j=1}^K \boldsymbol{t}_j^T + \boldsymbol{u}^T + \sum_{j=1}^K \boldsymbol{e}_{j,i_j}\right).
\end{aligned}
$$

Therefore, each device recovers $\mathsf{m} \otimes f_{\mathsf{prm}}\left(\sum_{j=1}^K \boldsymbol{e}_{j,i_j}^T\right)$.

Following the transformation (Definition 8) unforgeability is realized by adding a signature. We now discuss anonymity.

Anonymity from Dependency of Algebraic Systems:

We discuss the necessary number of colluded keys and authentication queries to distinguish the two commands $\mathsf{cmd}_{\mathcal{S}_0}$ and $\mathsf{cmd}_{\mathcal{S}_1}$ in the anonymity game (Definition 4). Assume that an attacker's strategy is to recover $\mathsf{vk}_{\mathsf{id}}$ or its alternative for some $\mathsf{id} \in \mathcal{S}_0 \triangle \mathcal{S}_1$ and try to verify $\mathsf{cmd}_{\mathcal{S}_b}$. Without loss of generality, we can let $\mathsf{id} = (1, 1, \ldots, 1)$.

Split the matrix W into two $M \times 2M$ matrices: $W = \begin{bmatrix} W_1 \\ W_2 \end{bmatrix}$. Denote $\boldsymbol{w}_{j,i}^T = W_1 \mathsf{dk}_{j,i}^T$ and $\boldsymbol{u}_{\mathsf{id}}^T = W_2 \mathsf{uk}_{\mathsf{id}}^T$. The verification key that one wants to recover is then written as $\mathsf{vk}_{\mathsf{id}}^T = (\boldsymbol{w}_{1,1}^T + \boldsymbol{u}_{\mathsf{id}}^T, \ldots, \boldsymbol{w}_{K,1}^T + \boldsymbol{u}_{\mathsf{id}}^T)$. $\mathsf{uk}_{\mathsf{id}}^T$ also satisfies $\sum_{j=1}^K A_{j,1} \mathsf{uk}_{\mathsf{id}}^T = \boldsymbol{u}^T$ for unknown matrices $A_{j,1}$ and unknown vector $\boldsymbol{u}^T$. Consider simultaneous equations to recover $\mathsf{vk}_{\mathsf{id}}$.

The number of variables to fix is KM for $\boldsymbol{w}_{j,1}$ $(j = 1, \ldots, K)$, $K \cdot rM$ for $A_{j,i}$, $2M^2$ for W_2 and r for $\boldsymbol{u}$. In addition, the number of unknown variables in $\mathsf{uk}_{\mathsf{id}}$ is r because the other $M - r$ variables can be random, i.e., have freedom, by construction. With a new colluded key $\mathsf{vk}_{\mathsf{id}'}^T$, less than MK independent equations can be obtained and it introduces new variables on $\boldsymbol{w}_{j,i}^T$ and $A_{j,i}$. An authentication query does not introduce new equations due to the random variables $\boldsymbol{t}_j$ in construction.

To minimize the number of unknown variables, when an attacker gets a new $\mathsf{vk}_{\mathsf{id}'}^T$, one can minimize the range of indexes. For $\mathsf{id} = (i_1, \ldots, i_N)$ that satisfies $i_j \in [2]$ for $j = 1, \ldots, s$ and $i_j = 1$ for $i = s+1, \ldots, K$, $t = 2^s - 1$ colluded keys are possible. The number of variables is $2(K+s)M$ for $\boldsymbol{w}_{j,i_j}^T$, $2(K+s) \cdot rM$ for A_{j,i_j}, $2M^2$ for W_2, tr for $\mathsf{uk}_{\mathsf{id}}^T$, r for $\boldsymbol{u}^T$. The total number of variables from the public key and t colluded key is $V = 2M^2 + 2(K+s)rM + 2(K+s)M + tr + r$, and the attacker has to solve them by tKM simultaneous equations. To fix the unique solution and $\mathsf{vk}_{\mathsf{id}}^T$, it is necessary to satisfy

$$
\begin{aligned}
&2M^2 + 2(K+s)rM + 2(K+s)M + tr + r < tKM \\
&\Leftrightarrow t > \frac{2M^2 + 2(K+s)rM + 2(K+s)M + r}{KM - r}.
\end{aligned}
$$

The last fraction is bounded by $M/K + 2r + 2$. Therefore, there is evidence of anonymity against $2 + 2r$ corruption. In the lattice-based instantiation described in the next section, r is the number

of samples in LWE and it is greater than 750. We believe that the security against the collusion of $2r = 1500$ devices is practically secure.

7 Concrete Scheme and Security Parameters

This section provides concrete schemes instantiated from the DLP and from the LWE problem, security parameters, and rough estimations of communication costs.

Below, we assume $N_j = 2$ for all j; thus, it is necessary to take $M \geq 2$ for the decryption keys. We use $M = 3$ because $M \geq 3$ is suitable for introducing some randomness in the keys.s The number of concatenated ABAs is $K = 20$ – 30 to control 10^6 – 10^9 devices in practice.

In addition, we discuss a relationship between the space of an ABA command and a cryptographic message for the practical use. Each target device can recover m whereas a nontarget device receives a random number which can be interpreted as a legitimate command. Because the message space in the DLP-based construction is exponential, the probability that such accidents occur is negligible. Meanwhile, a gimmick in a message is needed in the LWE-based construction to prevent such accidents.

7.1 DLP-Based Construction

We instantiate the DLP-based construction by setting $f_{\mathsf{prm}}(a) = g^a \bmod q$ to Definition 8 and 12.

Definition 13. • $\mathsf{Setup}(1^\lambda, K, \mathcal{D}) \to \mathsf{ak}$*: Fix a prime field* $\mathbb{Z}_q$ *and a dimension* M *of base MREs. Execute* $\mathsf{MRE}_j.\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{ek}_j, \{\mathsf{dk}_{j,i}^T\}_{i\in[N_j]})$ *for* $j \in [K]$ *where* $\mathsf{ek}_j := \{\mathsf{dk}_{j,i}^T\}$*. Generate random vectors* $\boldsymbol{a}_{j,i} \xleftarrow{\$} \mathbb{Z}_q^M$ *for* $j \in [K], i = 1, 2$*, a random invertible matrix* $W \in \mathbb{Z}_q^{2M\times 2M}$ *and a constant* $u \in \mathbb{Z}_q$*. Execute* $\Sigma.\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$ *The key is* $\mathsf{ak} = (\{\mathsf{dk}_{j,i}^T\}, \{\mathsf{ek}_j\}, \{\boldsymbol{a}_{j,i}\}, W, u, \mathsf{pk}, \mathsf{sk})$*.*
• $\mathsf{Join}(\mathsf{ak}, \mathsf{id}) \to \mathsf{vk}_{\mathsf{id}}$*: For a device* $\mathsf{id} = (i_1, i_2, \ldots, i_K)$*, generate a random vector* $\mathsf{uk}_{\mathsf{id}}$ *such that* $\sum_{j=1}^K \boldsymbol{a}_{j,i_j}\mathsf{uk}_{\mathsf{id}}^T = u \pmod q$*. Then, put the verification key by*

$$\mathsf{vk}_{\mathsf{id}} = \{(W(\mathsf{dk}_{1,i_1}||\mathsf{uk}_{\mathsf{id}})^T, \ldots, W(\mathsf{dk}_{K,i_K}||\mathsf{uk}_{\mathsf{id}})^T), \mathsf{pk}\}.$$

• $\mathsf{Auth}(\mathsf{ak}, \mathsf{m}, \mathcal{S}) \to \mathsf{cmd}_{\mathcal{S}}$*: Suppose the target devices are indicated by* $\mathcal{S}_1 \times \cdots \times \mathcal{S}_K \subset [2]^K$*. Select random numbers* $t_j \in \mathbb{Z}_q$ *and let* $x := t_1 + \cdots + t_K$*. Generate random vectors* $\boldsymbol{ct}_{j,i}$ *(*$j \in [K], i \in [2]$*) such that*

$$\boldsymbol{ct}_{j,i} \cdot \mathsf{dk}_{j,\ell}^T = \begin{cases} t_j & (i = \ell \text{ and } i \in S_j) \\ rand & (i \notin S_j) \end{cases}$$

where rand represents an output from random number generators except for t_j*.*

Define the vector $\boldsymbol{c}_{j,i} := (\boldsymbol{ct}_{j,i}^T||\boldsymbol{a}_{j,i})W^{-1}$ *and parse it into the coordinates by* $\boldsymbol{c}_{j,i} = (c_{j,i,1}, \ldots, c_{j,i,2M})$*. Then, the command* $\mathsf{cmd}_{\mathcal{S}}$ *is* $m \cdot g^{x+u}$ *and the sequence* $\{g^{c_{j,i,\ell}}\}$*. Finally, generate a signature to the command* $\Sigma.\mathsf{Sign}(\mathsf{sk}, \mathsf{cmd}_{\mathcal{S}}) \to \sigma$*. Ciphertext is the pair* $(\mathsf{cmd}_{\mathcal{S}}, \sigma)$*.*
• $\mathsf{Vrfy}(\mathsf{vk}_{\mathsf{id}}, (\mathsf{cmd}_{\mathcal{S}}, \sigma))$*: Check the signature by* $\Sigma.\mathsf{Vrfy}(\mathsf{pk}, \sigma, \mathsf{cmd}_{\mathcal{S}})$ *at first and if it returns* reject*, it stops with returning* reject*; otherwise, continue the process. For* $\mathsf{id} = (i_1, \ldots, i_K)$*, denote the vector part of* $\mathsf{vk}_{\mathsf{id}}$ *be* $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_K)$ *and let the* ℓ*-th element of* $\boldsymbol{v}_j$ *be* $v_{j,\ell}$*. For the command* $(m \cdot g^{x+u}, \{g^{c_{j,i,\ell}}\})$*, compute*

$$T_j = \prod_{\ell=1}^{2M} (g^{c_j, i_j, \ell})^{v_{j,\ell}} \text{ and } m \cdot g^{x+u} \cdot (T_1 \cdot \cdots \cdot T_K)^{-1}. \tag{7}$$

The proofs of correctness, anonymity and unforgeability are provided in the template construction. Below we estimate their communication and computation costs. The results are summarized in Table 1.

Table 1. Size and cost estimations in (EC)DLP settings. The table shows sizes of a verification key ($6K(\log_2 q)/8 + \mathsf{pksize}$ [Byte]) and command ciphertext ($((1 + 12K)(\log_2 q)/8 + \mathsf{sigsize}$ [Byte]). K, and q are the number of concatenated ABAs and a modulus to define finite fields, respectively. Following NIST standards, we assume $\mathsf{pksize} = N + 3L = 352$ [Byte] and $\mathsf{sigsize} = 2L = 64$ [Byte] in the DLP setting that uses a DSA. In addition, we assume $\mathsf{pksize} = 32$ [Byte] and $\mathsf{sigsize} = 64$ [Byte] in the ECDLP setting that uses EdDSA. The last two columns contain the expected timings in the million clocks of signature verification and cancelation computation to recover m in each target device with ARM Cortex-M4 processors.

	K	Size($\mathsf{vk_{id}}$) + pksize [Byte]	Size(cmd_S) +sigsize [Byte]	10^6 clocks (Σ.Vrfy)	10^6 clocks (Cancel)
Finite field	20	30720 + 352	61696+64	20.2	270,000
(112 bit security)	30	46080 +352	92416+64	26.6	405,000
Elliptic curve	20	3840 + 32	7712 + 64	3.1	66
(128 bit security)	30	5760 + 32	11552+64	3.9	99

Communication and storage costs Recall that M and q are the dimension of the base vector and a modulus to define finite fields, respectively. K is the number of concatenated base ABAs, which makes it possible to control 2^K devices. Let pksize and sigsize, respectively, be the sizes of the public key and signature; the central server has to keep them.

Following the specification of the NIST standard digital signature algorithm (NIST FIPS 186-4 [25] and 186-5[26]), we choose parameters DSA (N=2048, L=256) and EdDSA (Ed25519), which are expected to have at least 112 bit security. Concretely, in the DLP setting it takes $\lfloor \log_2 q \rfloor = 2048$ bits (size of the finite field) to store one element that derives $\mathsf{pksize} = N + 3L = 352$ [Byte] and $\mathsf{sigsize} = 2L = 64$ [Byte]. Also, in the ECDLP setting, it takes 256 bits to store the compressed representation of a point in Curve25519, which derives $\mathsf{pksize} = 32$ [Byte] and $\mathsf{sigsize} = 64$ [Byte].

The size of $\mathsf{vk_{id}}$ stored in each device is $2MK \log_2 q + \mathsf{pksize}$ [bits] because it consists K vectors of $2M$ dimension in $\mathbb{Z}_q$ and the public key of the signature.

Each command comprises $1+K\cdot 2\cdot 2M = 1+4KM$ numbers in $\mathbb{Z}_q$; thus, the communication cost is $(1+4KM)\log_2 q + \mathsf{sigsize}$. Concrete values obtained by setting $M = 3, K = 20, 30, \log_2 q = 2048$ are provided in Table 1.

Computational costs In the Join function, to generate $\mathsf{uk_{id}}$, we first randomly generate a vector $\boldsymbol{uk}'$ and compute $w = \sum_{j=1}^{K} \boldsymbol{a}_{j,i_j} \boldsymbol{uk} \pmod q$. We let $\mathsf{uk_{id}} = (u/w) \cdot \boldsymbol{uk}' \pmod q$ if $w \neq 0$; otherwise, repeat the sampling of $\boldsymbol{uk}'$. It takes approximately $MK(1 + 1/q)$ multiplications in $\mathbb{Z}_q$ and we neglect the $1/q$ element in the rest of the analysis. To compute $\mathsf{vk_{id}}$, it takes $4M^2K$ multiplications. Thus, Join requires approximately M samples and $MK + 4M^2K$ multiplications in $\mathbb{Z}_q$ to generate $\mathsf{vk_{id}}$ of one device.

In the Auth function, to generate $\boldsymbol{ct}_{j,i}$ for each j, randomly generate $\boldsymbol{ct}'_{j,i}$ and compute $w_{i,i} = \boldsymbol{ct}'_{j,i} \cdot \mathsf{dk}_{j,i}^T$. Then, let $\boldsymbol{ct}_{j,i} = (t_j/w_{i,i}) \cdot \boldsymbol{ct}'_{j,i}$ if $w_{i,i} \neq 0$. It requires $4MK$ multiplications in $\mathbb{Z}_q$. To

compute $\boldsymbol{c}_{j,i}$, it requires $8M^2K$ multiplications and the computation of cmd_S requires $1+4MK$ modular exponentiations, which requires almost $4MK\log_2 q$ multiplications in $\mathbb{Z}_q$ if we employ the binary method. To generate the signature, we have to compute a hash of $(1+4KM)\log_2 q$ length cmd_S.

In the Vrfy function, each device has to check the signature and compute $2MK$ modular exponentiations, which is roughly $2MK\log_2 q$ multiplications in $\mathbb{Z}_q$.

Below we provide the concrete expected timing using the Cortex-M4 processor.

Expected timing of verification The verification in each device requires signature verification and cancelation computation (7) to recover the message. We provide a rough estimation of processing times in the Cortex-M4 processor.

We assume that the DSA and EdDSA signatures are used in the DLP and ECDLP settings. These signature schemes are in the hash-and-sign paradigm that compute message hash. [29] reports Cortex-M4 can generate SHA-3 hash function 213 cycles/byte. For example, in our ECDLP setting with $K=20$, the length of the message to be signed is 7,712 [Byte] (Table 1) and 1.6 million cycles are required to compute the hash.

In addition to the hash computation, [13, Table 2] reports the verification of signature from very short message in Ed25519 DSA can be performed in 1.4 million cycles. Assuming that 2048-bit DSA is five times slower than DSA in low-resource environments [30].

The cancelation computation in (7) requires $2MK$ modular exponentiations (resp. scalar multiplications of points), $\mathbb{Z}_q$ in the DLP (resp. ECDLP) construction. An FFT-based implementation of 2048-bit modular exponentiation requires approximately 2,250 million cycles [5]. We estimated cancelation computation cost to be $4500\cdot MK$ million cycles in Table 1.

Meanwhile, point addition and squaring over the Curve25519 require only 548,873 cycles [36]. Thus, we use $1.1\cdot MK$ cycles as the cancelation computation cost in Table 1.

7.2 LWE-Based Construction

Recall that the LWE-based ABA is instantiated by setting $f_{\boldsymbol{p}}(\boldsymbol{x})=\boldsymbol{x}\boldsymbol{p}^T \bmod q$ to Definition 8 and 12. M and K are the parameters same as above.

Also, we use Q to the multiple of plaintext to avoid the effect of noises. Concretely, plaintext $\overline{\mathsf{m}}$ is an integer such that $0\le\overline{\mathsf{m}}<q/Q$ and is embedded it in the form of $\mathsf{m}=\overline{\mathsf{m}}\cdot Q$ in the command ciphertext. In verification, the device rounds the decoded message m' to $\overline{\mathsf{m}'}=\lfloor \mathsf{m}'/Q\rceil$. We use an integer L to distinguish legitimate and nonlegitimate commands. If the verification function returns $\overline{\mathsf{m}'}<L$, it is interpreted as a legitimate command and executed; otherwise, it returns the reject symbol. In our construction, we assume $q>2KLQ$ to separate $\overline{\mathsf{m}}$.

Definition 14. *(LWE-based Construction)*
• $\mathsf{Setup}(1^\lambda,K,\mathcal{D})\to\mathsf{ak}$: *Fix a prime field $\mathbb{Z}_q$ and a dimension M of base MREs. Execute* $\mathsf{MRE}_j.\mathsf{KeyGen}(\mathsf{pp})\to(\mathsf{ek}_j,\{\mathsf{dk}_{j,i}^T\}_{i\in[2]})$ *for $j\in[K]$, where* $\mathsf{ek}_j:=\{\mathsf{dk}_{j,i}^T\}$. *Generate random matrices* $A_{j,i}\xleftarrow{\$}\mathbb{Z}_q^{r\times M}$ *for $j\in[K],i\in[2]$, a random invertible matrix* $W\in\mathbb{Z}_q^{2M\times 2M}$ *and a vector* $\boldsymbol{u}^T\in\mathbb{Z}_q^r$. *Execute the key generation of signature* $\Sigma.\mathsf{KeyGen}(1^\lambda)\to(\mathsf{pk},\mathsf{sk})$. *The key is* $\mathsf{ak}=(\{\mathsf{dk}_{j,i}^T\},\{\mathsf{ek}_j\},\{A_{j,i}\},W,\boldsymbol{u}^T,\mathsf{pk},\mathsf{sk})$.
• $\mathsf{Join}(\mathsf{ak},\mathsf{id})\to\mathsf{vk}_{\mathsf{id}}$: *For a device* $\mathsf{id}=(i_1,\dots,i_K)$, *generate a random vector* $\mathsf{uk}_{\mathsf{id}}^T\in\mathbb{Z}_q^M$ *such that* $\sum_{j=1}^K A_{j,i_j}\mathsf{uk}_{\mathsf{id}}^T=\boldsymbol{u}^T \pmod q$. *The verification key is* $\mathsf{vk}_{\mathsf{id}}^T=\{(W(\mathsf{dk}_{1,i_1}^T||\mathsf{uk}_{\mathsf{id}}^T)^T,\dots,W(\mathsf{dk}_{K,i_K}^T||\mathsf{uk}_{\mathsf{id}}^T)^T),\mathsf{pk}\}$.
• $\mathsf{Auth}(\mathsf{ak},\mathsf{m}\in\mathcal{M},\mathcal{S})\to\mathsf{cmd}_{\mathcal{S}}$: *Suppose the target devices are indicated by* $\mathcal{S}_1\times\cdots\times\mathcal{S}_K\subset\prod[2]$. *Select random vectors* $\boldsymbol{t}_j^T\in\mathbb{Z}_q^r$ *so that* $\boldsymbol{p}\boldsymbol{t}_j^T\in\{LQ,\dots,2LQ-1\}$ *and let* $\boldsymbol{x}^T:=\boldsymbol{t}_1^T+\cdots+\boldsymbol{t}_K^T$. $\boldsymbol{p}\boldsymbol{x}^T$

is greater than KLQ because there is no overflow in $\mathbb{Z}_q$ by the condition $q > 2KLQ$. In addition, generate random matrices $CT_{j,i} \in \mathbb{Z}_q^{r\times M}$ $(j \in [K], i \in [2])$ such that

$$CT_{j,i} \cdot \mathsf{dk}_{j,\ell}^T = \begin{cases} \boldsymbol{t}_j^T + \boldsymbol{e}_{j,i}^T & (i = \ell \text{ and } i \in S_j) \\ \boldsymbol{z}_{j,i}^T & (i \notin S_j) \end{cases} \tag{8}$$

where $\boldsymbol{z}_{j,i}^T$ is a random vector such that $\boldsymbol{p}\boldsymbol{z}_{j,i}^T$ is less than LQ.

For each i, j, define the $2M$ column vectors $\boldsymbol{c}_{j,i,1}^T$ by $C_{j,i} := (CT_{j,i}||A_{j,i})W^{-1} = C_{j,i} = (\boldsymbol{c}_{j,i,1}^T||\cdots||\boldsymbol{c}_{j,i,2M}^T)$. The command $\mathsf{cmd}_{\mathcal{S}}$ is the pair of vector $\mathsf{vec} := (\mathsf{m}\cdot Q + \boldsymbol{p}(\boldsymbol{x}^T + \boldsymbol{u}^T), \{\boldsymbol{p}\boldsymbol{c}_{j,i,\ell}^T\})$ and its signature σ.

• $\mathsf{Vrfy}(\mathsf{vk}_{\mathsf{id}}, \mathsf{cmd}_{\mathcal{S}} = (\mathsf{vec}, \sigma))$: *Check the signature by $\Sigma.\mathsf{Vrfy}(\mathsf{pk}, \sigma, \mathsf{vec})$; if it is invalid, return* reject. *If otherwise, execute the decryption process as follows. For $\mathsf{id} = (i_1, \ldots, i_K)$, denote the vector part of $\mathsf{vk}_{\mathsf{id}}$ be $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_K)$ and let the ℓ-th element of $\boldsymbol{v}_j$ be $v_{j,\ell}$. For a command $(\mathsf{m} + \boldsymbol{p}(\boldsymbol{x}^T + \boldsymbol{u}^T), \{\boldsymbol{p}\cdot \boldsymbol{c}_{j,i,\ell}^T\} := \{F_{j,i,\ell}\})$, compute*

$$T_j = \sum_{\ell=1}^{2M} v_{j,\ell} F_{j,i_j,\ell} \text{ and } \overline{\mathsf{m}'} = \mathsf{m} + \boldsymbol{p}(\boldsymbol{x}^T + \boldsymbol{u}^T) - (T_1 + \cdots + T_K). \tag{9}$$

Decode the message by $\mathsf{m}' = \lfloor \overline{\mathsf{m}'}/Q \rceil$. If it is greater than L, return reject*; otherwise, return m'.*

The correctness and securities have already been discussed. We give details on the separation of legitimate commands. In the computation of $\overline{\mathsf{m}'}$, we have

$$\boldsymbol{p}(\boldsymbol{x}^T + \boldsymbol{u}^T) - (T_1 + \cdots T_K) = \boldsymbol{p}\boldsymbol{x}^T - \sum_{j=1}^{K} \boldsymbol{p}(CT_{j,i_j} \cdot \mathsf{dk}_{j,i_j}^T). \tag{10}$$

after canceling $\boldsymbol{u}^T$. Here, each factor is $\boldsymbol{p}(\boldsymbol{t}_j^T + \boldsymbol{e}_{j,i}^T)$ or $\boldsymbol{p}\boldsymbol{z}_{j,i}^T$ by (8). By the conditions $\boldsymbol{p}\boldsymbol{x}^T \geq KLQ$ and $\boldsymbol{p}\boldsymbol{z}_{j,i}^T < LQ$, if there is a factor from $\boldsymbol{p}\boldsymbol{z}_{j,i}^T$, the sum is greater than LQ and the resulting $\overline{\mathsf{m}'}$ is greater than L.

Parameter Restrictions From Theorem 1 (with the lattice instantiation), the hardness of the decision LWE problem with parameters (n, m, q, σ) is the security base of the anonymity of lattice-based ABA with parameters $M = 2n$, modulus q, and error parameter σ. We consider $(M = 2n, q, \sigma)$ for our ABA.

In addition to security, it should be limit the decoding error probability by changing q and Q. Following (10), the noise in the verification in a target device is $\sum_{j=1}^{K} \boldsymbol{p}\cdot\boldsymbol{e}_{j,i_j}^T$. Approximating each coordinate of $\boldsymbol{p}$ and $\boldsymbol{e}_{j,i_j}^T$ by continuous Gaussian $N(0, \sigma^2)$, the random variable that we consider is $\sum_{i=1}^{KM} \eta_i\eta_i'$ where all η_i and η_i' are independently sampled from the Gaussian. Its tail bound is estimated by

$$\Pr\left[\left|\sum_{i=1}^{KM} \eta_i\eta_i'\right| > \beta\right] < \frac{(\beta/\sigma^2)^{(KM-1)/2} K_{\frac{KM+1}{2}}(\beta/\sigma^2)}{2^{(KM-3)/2}\sqrt{\pi}\Gamma(KM/2)} \tag{11}$$

where $K_\nu(z)$ is the modified Bessel function of the second kind. Appendix A provides derivation. Of note,the numerical computation is easy due to the `kv` function in Python's `scipy.special` package. We set β such that the bound for the following situations.

The parameter should depend on the number of decoding processes for all systems and safety margins. For example, if a system controls 10^9 devices and works 100 years, and each device decodes a ciphertext in each second. Then, the number of decodings is $10^9 \cdot 3.2 \cdot 10^9 \approx 2^{61.5}$. With a safety margin 2^{64}, we can set β such that the right-hand side of (11) is less than 2^{-126}.

The aforementioned parameters may be conservative. If we assume that each decoding in devices occurs every 10 seconds, the working timespan is 10 years, and the safety margin is 2^{32}, the upper bound is 2^{-87}.

We provide the results of the two situations in Table 2 and 3. We can see the difference between required storages is less than 10%.

Communication and storage costs A verification key comprises $2KM = 4Kn$ elements of $\mathbb{Z}_q$ and a public key for signature verification. Thus, $\mathrm{Size}(\mathsf{vk}_{\mathsf{id}})$ is the smallest integer greater than $(4Kn \log_2 q)/8 + \mathsf{pksize}$.

A command ciphertext comprises $1 + 4KM = 1 + 8Kn$ elements of $\mathbb{Z}_q$ and a signature. Thus, $\mathrm{Size}(\mathsf{cmd}_S)$ is the smallest integer greater than $(1 + 8Kn)(\log_2 q)/8 + \mathsf{sigsize}$ [Byte].

Assuming the FALCON signature [10], $(\mathsf{pksize}, \mathsf{sigsize})$ is $(897, 666)$ and $(1793, 1280)$ for 128 and 256 bit securities, respectively. Based on the abovementioned restrictions on the parameters, we selected example parameters as follows.

Concrete parameters: As an example, we set $K = 20$ for controlling a million devices and set $\sigma = 3$. Let the legitimate message space be $L = 4$ (two bits). For LWE dimension n and $M = 2n$, set Q such that the right-hand side of (11) is less than 2^{-126}. and take q as the smallest prime number greater than $2KLQ$. We compute a practical bit security using Albrecht et al.'s [9] lattice estimator; the python code is provided in Appendix B. For example, $n = 785$, we obtain $Q = 34297$, $q = 5487539$, and bit security is 128.17, as described in the first line of 2.

Expected timing: The timing of signature verification is dominated by SHA-3 hash function comptation that requires 213 cycles/byte [29] whereas FALCON-512 and 1024 for a short message take less than 0.5 and 1 million cycles in Cortex-M4, respectively [18]. Meanwhile, the Chaskey hash function takes seven cycles/byte [24]. We provide the number of consumed cycles in 10^6 units for the SHA-3 and Chaskey situations in the 8th and 9th columns, respectively in Tables 2 and 3.

The cancelation computation (9) requires $2MK = 4nK$ additions and multiplications in $\mathbb{Z}_q$. For small q, this can be performed by a sequence of UMLAL instructions. For two 32-bit unsigned numbers $r0, r1$ and one 64-bit unsigned number $r2$, it computes $r0 \times r1 + r2$ and stores to a 64-bit register in one cycle [31]. Modular computations are unnecessary after the addition. Theoretically, it is sufficient to compute the modulo in every $2^{64}/q^2$ term. Hence, approximately $4nK$ cycles are required for the total computation. For a moderate q, divide the numbers in each 32-bit; we can use the schoolbook methods for multiplication and addition. Assuming q is less than 64-bit, the total computing time would be four times the cost of small q and it would require approximately $16nK$ cycles. In Table 2 (10th column), we assume that the computation takes $4nK$ and $16nK$ cycles for $L = 4$ and 256, respectively. The 10-th column of the tables provides summary.

8 Shorter Constructions and Nearly Weak Anonymity

In our template construction in Section 4.1, the vector dimension M of ciphertexts is fixed and must be larger than the number of devices N. This construction is inefficient when the number of targets $n = |\mathcal{S}|$ is much smaller than N.

Table 2. Example parameters of the LWE-based construction of ABA in a conservative situation. (Probability bound is 2^{-126}) The security level is computed by using Albrecht et al.'s [9] lattice estimator. K, L and Q are ABA parameter. n and $q > 2KLQ$ are LWE parameter. Size($\mathsf{vk}_{\mathsf{id}}$) and Size($\mathsf{cmd}_S$) are $(4Kn\log_2 q)/8 + \mathsf{pksize}$ and $(1+8Kn)(\log_2 q)/8 + \mathsf{sigsize}$, respectively. The last three columns comprise the expected timings in the million clocks of signature verification (w.r.t. SHA-3 and Chaskey hash functions) and cancelation computation to recover m in each target device with ARM Cortex-M4 processors.

Security Level	(K, L)	n	(q, Q)	Size($\mathsf{vk}_{\mathsf{id}}$) [Byte]	Size(cmd_S) [Byte]	10^6 cyc. (Σ.Vrfy) SHA-3	10^6 cyc. (Σ.Vrfy) Chaskey	10^6 cyc. (Cancel)
128.13	(20,4)	796	(6691207,41820)	183080+897	366160+666	78.47	3.04	0.07
128.12	(30,4)	831	(12553921,52308)	299160+897	598320+666	127.92	4.67	0.1
128.12	(20,256)	1036	(488407063,47696)	300440+897	600880+666	128.47	4.69	0.34
128.02	(30,256)	1070	(911539207,59345)	481500+897	963000+666	205.6	7.22	0.52
192.12	(20,4)	1186	(8164183,51026)	272780+1793	545560+1280	117.19	4.8	0.1
192.18	(30,4)	1237	(15312751,63803)	445320+1793	890640+1280	190.69	7.22	0.15
192.06	(20,256)	1530	(593346601,57944)	459000+1793	918000+1280	196.52	7.41	0.49
192.08	(30,256)	1580	(1107440647,72099)	734700+1793	1469400+1280	313.97	11.27	0.76
256.15	(20,4)	1560	(9361309,58508)	374400+1793	748800+1280	160.48	6.23	0.13
256.0	(30,4)	1624	(17542817,73095)	609000+1793	1218000+1280	260.42	9.51	0.2
256.0	(20,256)	2000	(678277123,66238)	600000+1793	1200000+1280	256.58	9.38	0.64
256.11	(30,256)	2065	(1265925121,82417)	960225+1793	1920450+1280	410.04	14.43	1.0

Table 3. Example parameters of the LWE-based construction of ABA in a relaxed margin situation. (Probability bound is 2^{-87})

Security Level	(K, L)	n	(q, Q)	Size($\mathsf{vk}_{\mathsf{id}}$) [Byte]	Size(cmd_S) [Byte]	10^6 cyc. (Σ.Vrfy) SHA-3	10^6 cyc. (Σ.Vrfy) Chaskey	10^6 cyc. (Cancel)
128.17	(20,4)	785	(5487539,34297)	180550+897	361100+666	77.39	3.01	0.07
128.12	(30,4)	820	(10299851,42916)	295200+897	590400+666	126.23	4.61	0.1
128.14	(20,256)	1025	(401223701,39182)	297250+897	594500+666	127.11	4.64	0.33
128.01	(30,256)	1059	(749030413,48765)	476550+897	953100+666	203.49	7.15	0.51
192.1	(20,4)	1170	(6697283,41858)	269100+1793	538200+1280	115.62	4.75	0.1
192.17	(30,4)	1221	(12566161,52359)	439560+1793	879120+1280	188.24	7.14	0.15
192.04	(20,256)	1514	(487516171,47609)	439060+1793	878120+1280	188.02	7.13	0.49
192.01	(30,256)	1564	(910141447,59254)	703800+1793	1407600+1280	300.8	10.84	0.76
256.02	(20,4)	1539	(7680017,48000)	353970+1793	707940+1280	151.77	5.94	0.13
256.07	(30,4)	1604	(14401459,60006)	577440+1793	1154880+1280	246.97	9.07	0.2
256.05	(20,256)	1980	(557455373,54439)	594000+1793	1188000+1280	254.03	9.3	0.64
256.02	(30,256)	2044	(1040394241,67734)	919800+1793	1839600+1280	392.82	13.86	0.99

After fixing the targets $\mathcal{S}$ and messages $\{m_i\}$, a minimal construction of the ciphertext in MRE (Definition 5) restricts a vector to the form of $\mathsf{ct} = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{|\mathcal{S}|+c}, 0, \ldots, 0)$ which has compressed representation of length $O(|\mathcal{S}|)$. Here, c is a constant that makes the ciphertext random for security. Based on this short-ciphertext MRE, we can construct an ABA template as in Definition 1 with small modifications. In particular, the last $\max(M - |\mathcal{S}| - c, 0)$ columns of the matrix CT are zero and the command is of the form $\mathsf{cmd}_{\mathcal{S}} = (m \otimes f_{\mathsf{prm}}(\boldsymbol{x}), F_1, \ldots, F_{|\mathcal{S}|+c})$. The correctness is straightforward. It does not have anonymity because the ciphertext length reveals the difference between $\mathsf{cmd}_{\mathcal{S}}$ and $\mathsf{cmd}_{\mathcal{S}'}$ when $|\mathcal{S}| \neq |\mathcal{S}'|$.

Instead, we can prove it has weak anonymity under the hardness assumption of the computational problem of distinguishing

$$(f_{\mathsf{prm}}(\boldsymbol{x}^T), f_{\mathsf{prm}}(\boldsymbol{c}_1^T), \ldots, f_{\mathsf{prm}}(\boldsymbol{c}_d^T)) \text{ and } (f_{\mathsf{prm}}(\boldsymbol{r}^T), f_{\mathsf{prm}}(\boldsymbol{c}_1^T), \ldots, f_{\mathsf{prm}}(\boldsymbol{c}_d^T)).$$

where $d = 1 + c$ corresponds to a situation in which the adversary chooses the target set such that $|\mathcal{S}| = |\mathcal{S}'| = 1$. This is the same problem except for the vector dimension of challenge instances in Definition 7.

The concatenation of the above ABA can be considered. Simple concatenation as in Definition 10 has a shorter ciphertext length $O(|\mathcal{S}_1| + \cdots + |\mathcal{S}_K|)$ and similar vulnerabilities related to the rearranging attacks.

8.1 Concatenated Template Construction

We propose a shorter ciphertext variant of the modified concatenated template construction (Definition 12). This variant reduces the ciphertext length, but the number of decryption keys in the controlled devices increases; thus it would be unsuitable for use in IoT devices. We believe that this construction implicitly suggests a transformation from a fixed-length anonymous ABA to a shorter ABA. The key ingredient is that each device has many decryption keys $\mathsf{vk}_{\mathsf{id},\tau}$ corresponding to the size $|\mathcal{S}_j|$ and uses the key adaptively.

Definition 15. *Fix global parameters: a prime field $\mathbb{Z}_q$, number of concatenations K, and the size N_i of i-th ABA. In addition, $M(\tau) \geq \tau + 1$ is a vector with the same dimension as the base MRE used for a target set size $|\mathcal{S}_j| = \tau$.*

• $\mathsf{Setup}(1^\lambda, K, \mathcal{D}) \to \mathsf{ak}$: *For each $j \in [K], \tau \in [N_j]$, execute $\mathsf{MRE}_{j,\tau}.\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{ek}_{j,\tau}, \{\mathsf{dk}_{j,\tau,i}^T\}_{i\in[N_j]})$ where $\mathsf{ek}_{j,\tau} := \{\mathsf{dk}_{j,\tau,i}^T\}_{i\in[N_j]}$. Generate random matrices $A_{j,\tau,i} \xleftarrow{\$} \mathbb{Z}_q^{r\times M_\tau}$ for $j \in [K], \tau \in [N_j], i$. Generate random invertible matrices $W_\tau \in \mathbb{Z}_q^{2M(\tau)\times 2M(\tau)}$ for $\tau = 1, \ldots, \max\{N_1, \ldots, N_K\}$. Generate a random vector $\boldsymbol{u}^T \in \mathbb{Z}_q^r$. The key is $\mathsf{ak} = (\{\mathsf{dk}_{j,\tau,i}^T\}, \{\mathsf{ek}_{j,\tau}\}, \{A_{j,\tau,i}\}, W_\tau, \boldsymbol{u}^T)$.*

• $\mathsf{Join}(\mathsf{ak}, \mathsf{id}) \to \mathsf{vk}_{\mathsf{id}}$: *For a device $\mathsf{id} = (i_1, i_2, \ldots, i_K)$, generate random vectors $\mathsf{uk}_{\mathsf{id},\tau} \in \mathbb{Z}_q^{1\times M(\tau)}$ satisfying $\sum_{j=1}^K A_{j,i_j,\tau}\mathsf{uk}_{\mathsf{id},\tau}^T = \boldsymbol{u}^T \pmod q$ for each τ. The set of verification keys is $\mathsf{vk}_{\mathsf{id},\tau} = \{W_\tau(\mathsf{dk}_{j,i_j,\tau}||\mathsf{uk}_{\mathsf{id},\tau})^T\}_{j\in[K],\tau\in[N_j]}$.*

• $\mathsf{Auth}(\mathsf{ak}, \mathsf{m}, \mathcal{S}) \to \mathsf{cmd}_{\mathcal{S}}$: *Suppose the target devices are indicated by $\mathcal{S}_1 \times \cdots \times \mathcal{S}_K \subset \prod[N_j]$. Pick random vectors $\boldsymbol{t}_j \in \mathbb{Z}_q^{1\times r}$ for $j \in [K]$ and let $\boldsymbol{x} := \boldsymbol{t}_1 + \cdots + \boldsymbol{t}_K$. Denote $\tau_j := |\mathcal{S}_j|$. Generate random matrices $CT_{j,i} \in \mathbb{Z}_q^{r\times M(\tau_j)}$ $(j \in [M], i \in [N_j])$ such that*

$$CT_{j,i} \cdot \mathsf{dk}_{j,\ell,M(\tau_j)} = \begin{cases} \boldsymbol{t}_j^T + \boldsymbol{e}_{j,i}^T & (i = \ell \text{ and } i \in S_j) \\ rand & (i \notin S_j) \end{cases}$$

where rand represents a random element far from $\boldsymbol{t}_j^T$. *Define the matrix* $C_{j,i} := (CT_{j,i}||A_{j,i,\tau_j})W_{\tau_j}^{-1} \in \mathbb{Z}_q^{r\times 2M(\tau_j)}$ *and split it into the* $2M(\tau_j)$ *column vectors by* $C_{j,i} = (\boldsymbol{c}_{j,i,1}^T||\cdots||\boldsymbol{c}_{j,i,2M(\tau_j)}^T)$. *Then, the command ciphertext* $\mathsf{cmd}_{\mathcal{S}}$ *is* $\mathsf{m}\otimes f_{\mathsf{prm}}(\boldsymbol{x}^T+\boldsymbol{u}^T)$ *and the sequence* $\{f_{\mathsf{prm}}(\boldsymbol{c}_{j,i,\ell}^T)\}_{j\in[K],i\in[N_j],\ell\in[2M(\tau_j)]}$.
• $\mathsf{Vrfy}(\mathsf{vk}_{\mathsf{id}},\mathsf{cmd}_{\mathcal{S}})$: *Denote* $\mathsf{vk}_{\mathsf{id}} = \{\boldsymbol{v}_{j,\tau}\}$ *according to its index. For* $\mathsf{id} = (i_1,\ldots,i_K)$, *denote the* ℓ*-th element of* $\boldsymbol{v}_{j,\tau_j}$ *be* $v_{j,\ell}$. *For the command* $(\mathsf{m}\otimes f_{\mathsf{prm}}(\boldsymbol{x}^T+\boldsymbol{u}^T),\{F_{j,i,\ell}\})$, *compute the sum* $T_j = \sum_{\ell=1}^{2M(\tau_j)} v_{j,\ell}\otimes F_{j,i_j,\ell}$ *and* $\mathsf{m}\otimes f_{\mathsf{prm}}(\boldsymbol{x}^T+\boldsymbol{u}^T)\otimes(T_1\otimes\cdots\otimes T_K)^{-1}$.

The proof of correctness is straightforward by replacing M with $M(\tau_j)$ for each j.

8.2 Componentwise Weak Anonymity

We discuss anonymity in the abovementioned construction. Because this construction limits the freedom of the target device selection, it does not have SA.

We provide an example to demonstrate that the concatenated short construction does not have weak anonymity. Consider the t-weak anonymity game (Definition 4) that the adversary's selection must satisfy $|\mathcal{S}| = |\mathcal{S}'|$. Assume an the ABA with $K = 3$ and $|N_j| = 8$ for all j. The adversary can select $\mathcal{S} = \mathcal{S}_1\times\mathcal{S}_2\times\mathcal{S}_3$ and $\mathcal{S}' = \mathcal{S}_1'\times\mathcal{S}_2'\times\mathcal{S}_3'$ such that $|\mathcal{S}_1| = |\mathcal{S}_2| = |\mathcal{S}_3| = 2$ and $|\mathcal{S}_1'| = 8, |\mathcal{S}_2| = |\mathcal{S}_3| = 1$. Because the number of columns of $C_{j,i}$ reveals $|\mathcal{S}_j|$, distinguishing is immediately performed without using any property of f_{prm}. Clearly, this attack does not require solving any algebraic equations discussed in Section 6.3.

To address this, we propose a tighter but somewhat artificial condition that we named componentwise weak anonymity.

Definition 16. *(Componentwise weak anonymity) Consider a componentwise variant of ABA such that all devices have* K*-dimensional* id. *In addition targets are selected via the products* $\mathcal{S}_1\times\cdots\times\mathcal{S}_K$ *with the rule that* $(i_1,\ldots,i_K)$ *is a target device if and only if* $i_j\in S_j$ *for all* j, *with the same protocol and advantages as* t*-anonymity (Definition 4). We say that it has componentwise weak* t*-anonymity if the advantage is negligible when the attacker's challenges* $\mathcal{S}_1 = \mathcal{S}_{1,1}\times\cdots\times\mathcal{S}_{1,K}$ *and* $\mathcal{S}_2 = \mathcal{S}_{2,1}\times\cdots\times\mathcal{S}_{2,K}$ *satisfy*

$$|\mathcal{S}_{1,j}| = |\mathcal{S}_{2,j}| \text{ for all } \ j. \tag{12}$$

The difference over weak t-anonymity is the only requirement for the challenge target sets. Because (12) implies (2), any componentwise ABA satisfying weak t-anonymity also satisfies componentwise weak t-anonymity. However, it is unclear the notion of componentwise weak anonymity matches a realistic situation.

In addition, with a similar argument in Section 6.3, solving algebraic equations can break the standard anonymity (Definition 4) and hence break the componentwise weak anonymity. Investigating relations among strengths of anonymity, ciphertext sizes, and possible format in the target device selection in practical applications is a problem to consider.

9 Concluding Remarks and Discussions

We proposed template constructions of ABA using several anonymity models. In particular, we showed that the concatenation of short ABAs can produce a practical ABA that can control N

devices with short ciphertext length $O(\log N)$. We provided instantiations from the DDH problem (non quantum-resilient) or the LWE problem (quantum-resilient).

The concatenated short construction is on the feasible area's borderline because of the trilemma of ABA construction and the notion of SA.

Practical issues To use ABA in practice to control numerous resource-limited devices, the following points should be considered. Investigation of the relationships among anonymity strengths, ciphertext sizes and freedom of target device selection is important. As we summarized in Section 1.2, several relationships are known under strict conditions. Practical relations should be considered under relaxed conditions.

In the LWE-based construciton in Section 7.2, relaxing safety margins (11) in decoding can change ciphertext lengths, although by a small amount. Optimization of such parameters would be relevant in practice.

Theoretical issues During the proof of anonymity security in template construction (Section 5.1) the security bases are DDH and LWE problems, which are instantiated for the template. Finding relations between them is an independent interest of the theory of ABA.

The concatenated construction in this study possibly implies a generic transformation from a fixed length ABA to a shorter length ABA. For example, suppose we have an ABA that controls n devices using an $O(n)$ length ciphertext. Assume that the number of devices is squared $N = n^2$ and index by $\mathsf{id} = (\mathsf{id}_1, \mathsf{id}_2) \in [n]^2$ for simplicity. Prepare ABA_i for $i = 1, 2$ and assume each ABA controls n devices. ABA_1 generates verification keys $\mathsf{vk}_{1,i}$ $(i \in [n])$ and transmits $\mathsf{vk}_{1,i}$ to devices (i, ℓ) for all $\ell \in [n]$. Similarly, ABA_2 generates $\mathsf{vk}_{2,j}$ $(j \in [n])$ and transmits it to devices (ℓ, j). Thus, device (i, j) has two verification keys $\mathsf{vk}_{i,j}^{(1)}$ and $\mathsf{vk}_{i,j}^{(2)}$.

For a single target $\mathcal{S} = \{(i, j)\}$, the command ciphertext is the concatenation of $ABA_1.\mathsf{Auth}(\mathsf{ak}, \mathsf{m}_1, \{i\})$ and $ABA_2.\mathsf{Auth}(\mathsf{ak}, \mathsf{m}_2, \{j\})$ whose length is $O(n) + O(n) = O(n)$. The device (i', j') can recover m_1 if $i' = i$ and m_2 if $j' = j$. Thus, only the device (i, j) can recover both messages and provide an ABA by adding some gimmicks to decode the true message from them. Considering the concatenation of command ciphertexts in the multi target situation $\mathcal{S} = \{(i_1, j_1), (i_2, j_2), \ldots\}$, we can obtain an ABA to control $N = n^2$ devices with $O(|\mathcal{S}|n)$ ciphertext length. A similar division into K-dimensional IDs can enable an ABA to control N devices with $O(|\mathcal{S}| \log N)$ cihpertext length. For small $|\mathcal{S}|$, this is practically efficient while also maintaining standard anonymity. Investigating the improvablity of $\log N$ is an interesting open problem.

Acknowledgement This research was in part conducted under a contract of "Research and development on new generation cryptography for secure wireless communication services" among "Research and Development for Expansion of Radio Wave Resources (JPJ000254)", which was supported by the Ministry of Internal Affairs and Communications, Japan. This work was in part supported by JSPS KAKENHI Grant Number JP22H03590.

References

[1] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. "Functional Encryption for Inner Product Predicates from Learning with Errors". In: *Advances in Cryptology – ASIACRYPT 2011*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 21–40. ISBN: 978-3-642-25385-0.

[2] Yoshinori Aono and Junji Shikata. "Anonymous Broadcast Authentication with Logarithmic-Order Ciphertexts from LWE". In: *Cryptology and Network Security - 22nd International Conference, CANS 2023, Augusta, GA, USA, October 31 - November 2, 2023, Proceedings*. Ed. by Jing Deng, Vladimir Kolesnikov, and Alexander A. Schwarzmann. Vol. 14342. Lecture Notes in Computer Science. Springer, 2023, pp. 28–50. DOI: 10.1007/978-981-99-7563-1_2. URL: https://doi.org/10.1007/978-981-99-7563-1_2.

[3] Elaine Barker et al. *Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography*. Website: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/nist.sp.800-56Ar3.pdf.

[4] Adam Barth, Dan Boneh, and Brent Waters. "Privacy in Encrypted Content Distribution Using Private Broadcast Encryption". In: *Financial Cryptography and Data Security*. Ed. by Giovanni Di Crescenzo and Avi Rubin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 52–64. ISBN: 978-3-540-46256-9.

[5] Hanno Becker et al. "Efficient Multiplication of Somewhat Small Integers Using Number-Theoretic Transforms". In: *Advances in Information and Computer Security*. Ed. by Chen-Mou Cheng and Mitsuaki Akiyama. Cham: Springer International Publishing, 2022, pp. 3–23. ISBN: 978-3-031-15255-9.

[6] Mihir Bellare et al. "Multirecipient Encryption Schemes: How to Save on Bandwidth and Computation Without Sacrificing Security". In: *IEEE Transactions on Information Theory* 53.11 (2007), pp. 3927–3943. DOI: 10.1109/TIT.2007.907471.

[7] Dan Boneh, Craig Gentry, and Brent Waters. "Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys". In: *Proceedings of the 25th Annual International Conference on Advances in Cryptology*. CRYPTO'05. Santa Barbara, California: Springer-Verlag, 2005, pp. 258–275. ISBN: 3540281142. DOI: 10.1007/11535218_16. URL: https://doi.org/10.1007/11535218_16.

[8] Arush Chhatrapati et al. "A Performance Evaluation of Pairing-Based Broadcast Encryption Systems". In: *Applied Cryptography and Network Security* 13269 (). URL: https://par.nsf.gov/biblio/10346776.

[9] *Estimate all the LWE, NTRU schemes!* Website: https://estimate-all-the-lwe-ntru-schemes.github.io/docs/.

[10] *Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU Specification v1.2 – 01/10/2020*. Website: https://falcon-sign.info/falcon.pdf.

[11] Nelly Fazio and Irippuge Milinda Perera. "Outsider-Anonymous Broadcast Encryption with Sublinear Ciphertexts". In: *Public Key Cryptography - PKC 2012 - 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012. Proceedings*. Ed. by Marc Fischlin, Johannes Buchmann, and Mark Manulis. Vol. 7293. Lecture Notes in Computer Science. Springer, 2012, pp. 225–242. DOI: 10.1007/978-3-642-30057-8_14. URL: https://doi.org/10.1007/978-3-642-30057-8_14.

[12] *FrodoKEM Learning With Errors Key Encapsulation Algorithm Specifications And Supporting Documentation (June 4, 2021)*. Website: https://frodokem.org/files/FrodoKEM-specification-20210604.pdf.

[13] Hayato Fujii and Diego F. Aranha. "Curve25519 for the Cortex-M4 and Beyond". In: *Progress in Cryptology – LATINCRYPT 2017*. Ed. by Tanja Lange and Orr Dunkelman. Cham: Springer International Publishing, 2019, pp. 109–127. ISBN: 978-3-030-25283-0.

[14] Sanjam Garg et al. "Building Efficient Fully Collusion-Resilient Traitor Tracing and Revocation Schemes". In: *Proceedings of the 17th ACM Conference on Computer and Communications Security*. CCS '10. Chicago, Illinois, USA: Association for Computing Machinery, 2010, pp. 121–130. ISBN: 9781450302456. URL: https://doi.org/10.1145/1866307.1866322.

[15] Robert E. Gaunt. "Inequalities for modified Bessel functions and their integrals". In: *Journal of Mathematical Analysis and Applications* 420.1 (2014), pp. 373–386. ISSN: 0022-247X. DOI: https://doi.org/10.1016/j.jmaa.2014.05.083. URL: https://www.sciencedirect.com/science/article/pii/S0022247X14005320.

[16] Adela Georgescu. "Anonymous Lattice-Based Broadcast Encryption". In: *Information and Communication Technology*. Ed. by Khabib Mustofa et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 353–362. ISBN: 978-3-642-36818-9.

[17] wolfies (https://math.stackexchange.com/users/74360/wolfies). *Distribution of sum of product-normal distributions.* Mathematics Stack Exchange. URL:https://math.stackexchange.com/q/1113150 (version: 2017-07-22). eprint: https://math.stackexchange.com/q/1113150. URL: https://math.stackexchange.com/q/1113150.

[18] Matthias J. Kannwischer et al. *pqm4: Testing and Benchmarking NIST PQC on ARM Cortex-M4*. Workshop Record of the Second PQC Standardization Conference. https://cryptojedi.org/papers/\#pqm4. 2019.

[19] Aggelos Kiayias and Katerina Samari. "Lower Bounds for Private Broadcast Encryption". In: *Information Hiding*. Ed. by Matthias Kirchner and Dipak Ghosal. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 176–190. ISBN: 978-3-642-36373-3.

[20] Hirokazu Kobayashi et al. "Tight lower bounds and optimal constructions of anonymous broadcast encryption and authentication". In: *Designs, Codes and Cryptography* (Apr. 2023), pp. 1–40. DOI: 10.1007/s10623-023-01211-x.

[21] Kaoru Kurosawa et al. "Some Bounds and a Construction for Secure Broadcast Encryption". In: *Advances in Cryptology - ASIACRYPT '98*. Ed. by Kazuo Ohta and Dingyi Pei. Vol. 1514. Lecture Notes in Computer Science. Springer, 1998, pp. 420–433.

[22] Jiwon Lee et al. "Combinatorial Subset Difference – IoT-Friendly Subset Representation and Broadcast Encryption". In: *Sensors* 20.11 (2020). ISSN: 1424-8220. DOI: 10.3390/s20113140. URL: https://www.mdpi.com/1424-8220/20/11/3140.

[23] Benoît Libert, Kenneth G. Paterson, and Elizabeth A. Quaglia. "Anonymous Broadcast Encryption: Adaptive Security and Efficient Constructions in the Standard Model". In: *Public Key Cryptography – PKC 2012*. Ed. by Marc Fischlin, Johannes Buchmann, and Mark Manulis. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 206–224. ISBN: 978-3-642-30057-8.

[24] Nicky Mouha et al. "Chaskey: An Efficient MAC Algorithm for 32-bit Microcontrollers". In: *Selected Areas in Cryptography – SAC 2014*. Ed. by Antoine Joux and Amr Youssef. Cham: Springer International Publishing, 2014, pp. 306–323. ISBN: 978-3-319-13051-4.

[25] National Institute of Standards and Technology (NIST). *Digital Signature Standard (DSS)*. Website: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf.

[26] National Institute of Standards and Technology (NIST). *Digital Signature Standard (DSS)*. Website: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf.

[27] Attrapadung Nuttapong. "Unified Frameworks for Practical Broadcast Encryption and Public Key Encryption with High Functionalities". PhD thesis. 2007.

[28] Oded Regev. "On Lattices, Learning with Errors, Random Linear Codes, and Cryptography". In: *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*. STOC '05. Baltimore, MD, USA: Association for Computing Machinery, 2005, pp. 84–93. ISBN: 1581139608. DOI: 10.1145/1060590.1060603. URL: https://doi.org/10.1145/1060590.1060603.

[29] Rajeev Sobti and Geetha Ganesan. "Performance Evaluation of SHA-3 Final Round Candidate Algorithms on ARM Cortex-M4 Processor". In: 12.1 (2018), pp. 63–73. ISSN: 1930-1650. DOI: 10.4018/IJISP.2018010106. URL: https://doi.org/10.4018/IJISP.2018010106.

[30] *Software Crypto Benchmark Comparison STM32F4 vs STM32H5*. https://www.oryx-embedded.com/benchmark/st/compare-stm32f4-vs-stm32h5.html.

[31] *https://developer.arm.com/documentation/ddi0439/b/CHDDIGAC*.

[32] Yohei Watanabe, Naoto Yanai, and Junji Shikata. "Anonymous Broadcast Authentication for Securely Remote-Controlling IoT Devices". In: *Advanced Information Networking and Applications*. Springer International Publishing, 2021, pp. 679–690.

[33] Yohei Watanabe, Naoto Yanai, and Junji Shikata. *IoT-REX: A Secure Remote-Control System for IoT Devices from Centralized Multi-Designated Verifier Signatures*. 2022. arXiv: 2208.03781 [cs.CR].

[34] Eric W. Weisstein. *Modified Bessel Function of the Second Kind*. https://mathworld.wolfram.com/ModifiedBesselFunctionoftheSecondKind.html.

[35] Eric W. Weisstein. *Normal Product Distribution*. https://mathworld.wolfram.com/NormalProductDistribution.html.

[36] "X25519-Cortex-M4". URL: https://github.com/Emill/X25519-Cortex-M4.

A On the Sum of Independent Gaussian Products

In this section, we estimate the probability $\Pr\left[|\sum_{i=1}^{m} e_i e_i'| > \beta\right]$, where e_i and e_i' are independently sampled from the continuous Gaussian distribuntion $N(1,0)$. The key idea for computing the probability density function is from [17].

Let e, e' be drawn from $N(0,1)$ independently. The probability density function of the product $e \cdot e'$ is thus given by $f_1(z) = \dfrac{K_0(|z|)}{\pi}$. (See, e.g., [35].) Here

$$K_\nu(z) = \frac{\Gamma(\nu + 1/2)(2z)^\nu}{\sqrt{\pi}} \int_0^\infty \frac{\cos t}{(t^2 + z^2)^{\nu+1/2}} dt$$

is the modified Bessel function of the second kind [34].

The characteristic function is

$$\varphi_1(t) = \int_{-\infty}^{\infty} e^{itz} f_1(z) dz = \frac{2}{\pi} \int_0^\infty K_0(z) \cos tz dz = \frac{1}{\sqrt{1+t^2}}.$$

Thus, the characteristic function of the sum distribution is $\varphi_n(t) = (\varphi_1(t))^n = (1+t^2)^{-n/2}$. Throughout the inversion formula, the probability density function is

$$f_n(z) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-itz} \varphi_n(t) dt = \frac{1}{\pi} \int_0^\infty \frac{\cos tz}{(1+t^2)^{n/2}} dt = \frac{|z|^{(n-1)/2} K_{\frac{n-1}{2}}(|z|)}{2^{(n-1)/2} \sqrt{\pi} \Gamma(n/2)}.$$

With formula for an upper bound (see, for example [15, Th. 2.5])

$$\int_x^\infty t^\nu K_\nu(t)dt < x^\nu K_{\nu+1}(x),$$

the tail probability is bounded by

$$\Pr\left[\left|\sum_{i=1}^m e_i e_i'\right| > \beta\right] < \frac{\beta^{(n-1)/2} K_{\frac{n+1}{2}}(\beta)}{2^{(n-3)/2}\sqrt{\pi}\Gamma(n/2)}.$$

B Parameter Generating Codes

We provide our python code to generate parameters in Tables 2 and 3 in Section 7.2.

For example, execute it with options `-probpower -126 -L 4 -K 20 -lwesigma 3.0 -output ABAconservative.txt` will generate the 1,5 and 9-th lines in Table 2.

```
from sage.all_cmdline import *   # import sage library
import math
import sympy
from estimator import *
import mpmath
import argparse

def getbetabyproblimit(lwesigma,KM,prbound):
  #Get probability bound Pr[b>\sum e_i*e'_i] = prbound
  #where e_i and e'_i's are independent Gaussian with derivation sigma^2
  #and the sum is over i=1,...,KM
  b=KM
  phase=0
  logprbound = log(prbound)
  mt = 10**5
  while True:
    logbound=((KM-1)/2) * log(b/lwesigma/lwesigma)
    logbound += log( mpmath.besselk((KM+1)/2,1.0*b/lwesigma/lwesigma, maxterms=mt))
    logbound -= (KM-3)/2*log(2.0) + 0.5*log(math.pi)
    logbound -= log(mpmath.gamma(KM/2))
    #print(b,logbound,logprbound)
    if phase==0:
      if logbound < logprbound:
        phase = 1
        step = b/4
        b = b/2
      else:
        b*=2
    else: #phase==1
      if logbound < logprbound:
        b-=step
      else:
        b+=step
      step /= 2
      if step < 0.1:
        break
  return b

def makeparamtable(K,L,prbound,startn,lwesigma,output):
  lwen = startn
  f = open(output, 'a')
  while True:
    M=2*lwen
    lQ = ceil(getbetabyproblimit(lwesigma,K*M,prbound)*2)
    lweq = sympy.nextprime(2*K*L*lQ, 1)
```

```
46      ABA128=LWE.Parameters(n=lwen, q=lweq, Xs=ND.DiscreteGaussian(lwesigma), Xe=ND.
            DiscreteGaussian(lwesigma), m=10*lwen, tag='ABA128')
47      result = LWE.primal_bdd(ABA128)
48      cost = math.log2(result.rop)  #bit security
49      pksize = "+897" #size of FALCON512 key (128bit)
50      sigsize = "+666"  #size of FALCON512 signature (128bit)
51      falconcycles = 473964 #FALCON512
52      if cost > 190:
53        pksize = "+1793"  #size of FALCON1024 key (256bit)
54        sigsize = "+1280" #size of FALCON1024 signature (256bit)
55        falconcycles = 978558 #FALCON1024
56      commandsize = ceil(8*K*lwen * ceil( log(lweq)/log(2))/8)
57      vrfycycles = 213 * commandsize + falconcycles #verification with SHA-1
58      vrfycycles2 = 7 * commandsize + falconcycles  #verification with Chaskey
59      cancelcycles = 4*lwen*K
60      if L==256:
61        cancelcycles = 16*lwen*K
62
63      print("--------------------")
64      print("Q=",lQ, "lweq=",lweq)
65      print("log2cost=", floor(cost*100)/100.0)
66      print("n=", lwen, "␣(K,L)=(",K, "," , L ,")␣(q,Q)=(",lweq, "," ,lQ,")", end="␣")
67      print( "pksize=" ,ceil(4*K*lwen * ceil( log(lweq)/log(2))/8) + int(pksize) , end="␣")
68      print( "sigsize=" , ceil(8*K*lwen * ceil( log(lweq)/log(2))/8) + int(sigsize) , end="␣")
69      print( "vrfycycles(SHA-3)[/10^6]=", ceil(vrfycycles/10000)/100.0 , end="␣")
70      print( "vrfycycles(Chaskey)[/10^6]=", ceil(vrfycycles2/10000)/100.0 , end="␣")
71      print( "cancelcycles[/10^6]=", ceil(cancelcycles/10000)/100.0 )
72
73      #tex style output for tables
74      print(floor(cost*100)/100.0,"&(", K, "," , L ,")&" , lwen , "&(",lweq, "," ,lQ,")&" , end='
            ',  file=f)
75      print( ceil(4*K*lwen * ceil( log(lweq)/log(2))/8) , pksize , "&" , end='',  file=f)
76      print( ceil(8*K*lwen * ceil( log(lweq)/log(2))/8) , sigsize , "&" , end='',  file=f)
77      print( ceil(vrfycycles/10000)/100.0 , "&" , end='',  file=f)
78      print( ceil(vrfycycles2/10000)/100.0 , "&" , end='',  file=f)
79      print( ceil(cancelcycles/10000)/100.0 , "\\\\",  file=f)
80
81      #choose next n
82      if 127 < cost < 128 or 191 < cost < 192 or 255 < cost < 256:
83        lwen += 1
84      elif 110 < cost < 128 or 182 < cost < 192 or 246 < cost < 256:
85        lwen += 5
86      else:
87        lwen += 50
88      if cost > 260:
89        break
90    return
91
92  parser = argparse.ArgumentParser(description='ABA␣parameter')
93  parser.add_argument('--probpower',action='store',default=-126)  #probability of
94  parser.add_argument('--L',action='store',default=4) #size of message space y of
95  parser.add_argument('--K',action='store',default=20)  #Control 2^K devices
96  parser.add_argument('--lwesigma',action='store',default=3.0)  #scaling parameter of LWE
97  parser.add_argument('--output',action='store',default="ABAparam.txt") #output filename
98  args = parser.parse_args()
99  prbound = 2**(int(args.probpower))
100
101 K=int(args.K)
102 L=int(args.L)
103 lwesigma = float(args.lwesigma)
104 makeparamtable(K,L,prbound,400,lwesigma,args.output)
```