CRYPTOGRAPHIC SCHEMES WITH MINIMUM DISCLOSURE OF PRIVATE INFORMATION IN ATTRIBUTE-BASED ENCRYPTION AND MULTIPARTY COMPUTATION

TAKASHI NISHIDE

THE UNIVERSITY OF ELECTRO-COMMUNICATIONS

SEPTEMBER 2008

CRYPTOGRAPHIC SCHEMES WITH MINIMUM DISCLOSURE OF PRIVATE INFORMATION IN ATTRIBUTE-BASED ENCRYPTION AND MULTIPARTY COMPUTATION

TAKASHI NISHIDE

THE UNIVERSITY OF ELECTRO-COMMUNICATIONS

GRADUATE SCHOOL OF ELECTRO-COMMUNICATIONS

A DISSERTATION SUBMITTED FOR
THE DOCTOR OF PHILOSOPHY IN ENGINEERING

SEPTEMBER 2008

CRYPTOGRAPHIC SCHEMES WITH MINIMUM DISCLOSURE OF PRIVATE INFORMATION IN ATTRIBUTE-BASED ENCRYPTION AND MULTIPARTY COMPUTATION

SUPERVISORY COMMITTEE

CHAIRPERSON: PROFESSOR KAZUO OHTA

MEMBER: PROFESSOR KINGO KOBAYASHI

MEMBER: PROFESSOR KIYOSHI ANDO

MEMBER: PROFESSOR TETSURO NISHINO

MEMBER: ASSOCIATE PROFESSOR NOBORU KUNIHIRO

属性ベース暗号とマルチパーティ計算におけるプライベート情報の開示を最小にした暗号方式

西出 隆志

論文概要

暗号プロトコルがプライバシー保護/強化技術の要素技術として活用されるようになり，従来，プライバシーの安全性の問題から実現できなかったサービスがネットワーク上で実現可能となりつつある．

本論文では，属性ベース暗号 (Attribute-Based Encryption) とマルチパーティ計算 (Multiparty Computation) のプライバシー保護技術について検討する．

属性ベース暗号では，プライバシー強化の観点から，有用な情報を含みうる復号条件をも復号者に対して秘匿可能とし，復号条件の表現能力においても既存方式に比べより有用な性質を持つ方式を提案する．

マルチパーティ計算では，効率の面から実用的とは言えなかった既存方式を改良し，秘密入力の比較，同一性判定，区間判定という基本構成要素に対し計算量，通信回数がより効率的な方式を提案する．併せて，分散鍵生成プロトコルによって閾値準同型 Paillier 暗号に基づいたマルチパーティ計算を信頼できる第三者機関を想定せずに構成する方式も提案する．

CRYPTOGRAPHIC SCHEMES WITH MINIMUM DISCLOSURE OF PRIVATE INFORMATION IN ATTRIBUTE-BASED ENCRYPTION AND MULTIPARTY COMPUTATION

TAKASHI NISHIDE

ABSTRACT

Modern cryptography can serve as a building block for privacy preserving and enhancing technologies and it enables us to realize various kinds of online services on the network which cannot exist unless privacy issues are solved.

In this dissertation, we focus on two technologies called Attribute-Based Encryption (ABE) and Multiparty Computation (MPC).

We propose two ABE schemes where an encryptor can hide a decryption policy that can contain sensitive information in terms of privacy enhancement and the expressiveness of the proposed schemes has more desirable properties compared with the existing schemes.

In MPC, we improve the efficiency of the existing scheme and propose more efficient protocols, in terms of round and communication complexities, for comparison, equality, and interval tests of secret inputs which are integer arithmetic primitives. In addition, we propose an MPC protocol for distributed key generation of the threshold Paillier cryptosystem without a trusted third party.

Contents

Chapter 1

Introduction

Privacy is a problem. With the widespread use of computers, various kinds of data including private information are digitalized and stored in the computers. Furthermore, such data are accessed or shared through the network or the Internet. Once the private data leak out and spread across the network, it is virtually impossible to recover the data. Thus, the more open computerized environments we have, the more problems related to privacy we have.

On the other hand, private information is valuable. If we can utilize private information with the minimum disclosure of it when necessary, it will be useful for the real-world applications. Modern cryptography can serve as building blocks for solving online privacy and security problems. Actually it is not only about hiding information but also about secure information sharing and privacy-enhancing techniques. In additon to its theoretical beauty, it can have a practical impact on the real world and has potential to change the way people interact in the digital world.

In this dissertation, we consider two main subjects among privacy applications. One is *Attribute-Based Encryption* (ABE) and the other is *Multiparty Computation* (MPC).

In order to protect private data, it is crucial for us to have control over who can access important or sensitive data. Typically an access control policy on digital data is enforced by a software mechanism, but the server computers where the data are stored might be compromised by malicious outsiders and then we will lose control of privacy.

Obviously data encryption can alleviate the privacy concerns. However, if the encrypted data must be shared among multiple people in an appropriate way, the management of cryptographic

keys becomes complex. If the data holder (i.e., the encryptor) encrypts her file for each potential decryptor she wants to share the file with, she must store multiple encrypted files for one file on the server and it will require huge data storage. On the other hand, the encryptor can share one key per file with potential decryptors and store one encrypted file per file on the server, but this will also lead to the complex key management for both the encryptors and decryptors because keys must be generated and shared with the decryptors each time files are created. Furthermore the encryptor may not be able to know the identities of the potential decryptors when she encrypts her files in the distributed setting. Therefore, it is a non-trivial task to share encrypted data with the simple key management and this is where ABE comes in.

ABE enables us to share encrypted data with encryptor-specified access control policies and the policies are described with attributes of people rather than identities of people. Thus the encrypted data are not intended for specific individuals and ABE can enforce attribute-based access control on encrypted data in a cryptographic way. In this dissertation, we consider hiding even encryptor-specified access control policies for privacy enhancement. Suppose Alice is looking for her partner who matches the criteria she specified. Then she can broadcast her message encrypted by ABE to people and the people who could decrypt the message can contact her. In such a situation she might want to hide her criteria also in addition to the message because of privacy concerns. In Chapter 2, we deal with such ABE and construct ABE schemes where the encryptor can hide the access control policies with more flexible properties compared with the existing schemes. The results in Chapter 2 were published as [NYO08].

In MPC, we consider secure data sharing. Usually sensitive data are valuable and if we can collect sensitive data and extract meaningful information by computing some agreed function without revealing the individual sensitive data, it may be beneficial to the individual data holders. MPC enables a set of parties with private inputs to jointly compute an agreed function of their inputs and obtain only the output without revealing the private inputs. By utilizing MPC, we can realize secure computation without relying on a trusted third party which can be a single point of failure. For example, in the famous problem called Yao's millionaires' problem [Yao82], two millionaires want to know who is richer without revealing their wealth and this can be solved by using MPC. Though theoretical results (e.g., [Yao86, GMW87, BGW88, CCD88]) show any

function can be computed securely without revealing the private inputs, the efficiency of such general schemes is not good enough for practical use. Therefore, improving the efficiency by constructing protocols for specific use is meaningful and necessary. For example, in [BCD+08], an MPC protocol for the specific auction system was implemented and tested and it showed the viability of large-scale MPC for a real-world application. In this dissertation, we focus on MPC for integer arithmetic and building on [DFK+06], construct MPC protocols for comparison, equality, and interval tests of secret data, which are important building blocks. We discuss these protocols in Chapter 3 that are the results of [NO07]. In our protocols, the parties participating in MPC do not need to know any private inputs and can serve as computing agents for input holders (as *the client-server model* in [DI05]). This means the following. In the Yao's millionaires' problem, each input holder (millionaire) knows his private input value and the input holders perform MPC. However, in our protocols all the private input values have only to be given by the input holders and shared among the parties participating (which we can call servers) in MPC and the servers can perform MPC without knowing any private inputs. This computing model is very general and the input holders can request the servers to perform MPC with their private inputs, which can be considered as a kind of online service provided by the servers performing MPC. In such a scenario, secret data are always distributed and the risk of data leakage is reduced.

In Chapter 4, we also consider an MPC protocol for distributed key generation of the threshold Paillier cryptosystem [FPS00]. Typically MPC is realized by using Shamir secret sharing [Sha79] or threshold homomorphic cryptosystems and the Paillier cryptosystem is one of the most used homomorphic cryptosystems. The distributed generation of an RSA modulus (needed also for the Paillier cryptosystem) is a non-trivial and time-consuming task and often MPC based on the threshold Paillier cryptosystem assumes that the public and secret keys are generated by a trusted third party or a secure hardware box. Building on [BF97, FS01, ACS02], we show how the MPC protocols for generating an RSA modulus in a distributed way can be adapted to the case of the threshold Paillier cryptosystem.

Chapter 2

Privacy Enhancing Attribute-Based Encryption

In this chapter, we introduce attribute-based encryption schemes with a privacy enhancing functionality. We propose attribute-based encryption schemes where encryptor-specified access structures (also called ciphertext policies) are hidden. By using our schemes, an encryptor can encrypt data with a hidden access structure. A decryptor obtains her secret key associated with her attributes from a trusted authority in advance and if the attributes associated with the decryptor's secret key do not satisfy the access structure associated with the encrypted data, the decryptor cannot decrypt the data or guess even what access structure was specified by the encryptor. We prove security of our construction based on the Decisional Bilinear Diffie-Hellman assumption and the Decision Linear assumption. In our security notion, even the legitimate decryptor cannot obtain the information about the access structure associated with the encrypted data more than the fact that she can decrypt the data. The results in this chapter were published as [NYO08].

2.1 Introduction

2.1.1 Background

In the distributed setting, we need to enforce access control polices to protect various resources. In such settings, it may be suitable to specify access control policies based on attributes rather than individual identities, because an identity may not have enough information about its entity. Attribute-based encryption (ABE) is a mechanism by which we can realize such access control in a cryptographic way. There are two kind of ABE schemes, key-policy and ciphertext-policy ABE schemes.

In the key-policy ABE schemes [GPS+06, OSW07, SW05, KSW08], ciphertexts are associated with sets of attributes and users' secret keys are associated with access structures. If the attributes associated with the ciphertext satisfy the access structure of the secret key, the secret key holder can decrypt the ciphertext successfully. Also the concept of searchable and predicate encryption [BW07, SBC+07] is related to key-policy ABE in the sense that successful decryption is conditional on access structure associated with secret keys.

On the other hand, in the ciphertext-policy ABE (CP-ABE) schemes [BSW07, CN07, KSW08, LS08], the situation is reversed. That is, attributes are associated with secret keys and access structures are associated with ciphertexts and called ciphertext policies. The access structures are described with the attributes and therefore the concept of CP-ABE is closely related to Role-Based Access Control.

In this work, we focus on CP-ABE and construct a CP-ABE scheme where we can hide encryptor-specified access structures associated with ciphertexts. Our scheme can be considered as a recipient-anonymous targeted broadcast and the relation of our scheme to a normal CP-ABE scheme is similar to that of anonymous identity-based encryption (IBE) to normal IBE. For example, suppose a company wants to hire certain qualified people who satisfy the policy the company specified and the policy may contain the useful information about the company's business strategy. The company can post a message encrypted by our CP-ABE scheme on a public bulletin board to seek applications. By doing so, the company can keep the important policy confidential. Since

the policy is hidden, the rival companies cannot know what kind of policy the company used to hire its employees.

In the ABE schemes, *collusion-resistance* is an important property. We do not want the secret key holders to be able to combine their secret keys to decrypt ciphertexts neither of them can decrypt. By building on the previous schemes [BSW07, CN07], we can also realize collusion-resistant CP-ABE schemes.

2.1.2 Our Contributions

We construct two CP-ABE schemes with partially hidden ciphertext policies in the sense that possible values of each attribute in the system should be known to an encryptor in advance and the encryptor can hide what subset of possible values for each attribute in the ciphertext policy can be used for successful decryption. In our schemes, encryptors can use wildcards to mean that certain attributes are not relevant to the ciphertext policy in a hidden way. The security proof of our first construction is given under the Decisional Bilinear Diffie-Hellman assumption and the Decision Linear assumption. Since these assumptions are general, we can use a large variety of elliptic curves (including both asymmetric and symmetric bilinear pairings) to implement our first scheme though we use the symmetric notation for ease of exposition. The security proof of our second construction is given in the generic group model and the second construction needs DDH-hard groups, but with a property inherited from [BSW07], the second construction is more flexible than the first construction in that new attributes can be added in the ciphertext policy securely with the existing public parameters being unchanged even after the system setup is done. We mention this aspect in Sect. 2.7 in more detail. We describe our constructions in the multi-valued attribute setting where an attribute can take multiple values and this setting is a generalization of the access structures used in [CN07]. In our security notion, even the legitimate decryptor cannot obtain the information about the ciphertext policy more than the fact that she can decrypt the data.

2.1.3 Related Work

Bethencourt, Sahai, and Waters [BSW07] proposed the first CP-ABE scheme. Their scheme allows the ciphertext policies to be very expressive, but the security proof is in the generic group model and the policies need to be revealed in the ciphertexts because decryptors must know how they should combine their secret key components for decryption. Cheung and Newport [CN07] proposed a provably secure CP-ABE scheme and their scheme deals with negative attributes explicitly and supports wildcards in the ciphertext policies but the policies need to be revealed as in [BSW07]. Kapadia, Tsang, and Smith [KTS07] also proposed a CP-ABE scheme and their scheme realizes hidden ciphertext policies that have the same expressiveness as [CN07], but their scheme is not collusion-resistant and needs an online semi-trusted server that must know the attributes' values every user in the system has and re-encrypt ciphertexts for each user when the user retrieves the ciphertexts. Such an online semi-trusted server can be a performance bottleneck in the system while, in our schemes, encryptors can just post or broadcast ciphertexts. Lubicz and Sirvent [LS08] proposed another CP-ABE scheme that has the same expressiveness as [CN07] and only 3 pairing computations are needed for decryption, but the ciphertext policies need to be revealed for decryption. Shi et al. [SBC+07] proposed a predicate encryption scheme that focuses on range queries over huge numbers, the dual of which can also realize a CP-ABE scheme where an encryptor can specify a number range in the ciphertext policy. The security proof of [SBC+07] is based on the security notion weaker than ours, which is called *match-revealing security* in [SBC+07] and the number of attributes must be small because the decryption cost is exponential in the number of attributes. Boneh and Waters [BW07] proposed a predicate encryption scheme based on the primitive called *Hidden Vector Encryption* or HVE for short. The scheme in [BW07] can realize the same functionality as ours by using the opposite semantics of subset predicates described in [BW07] (see Sect. 2.2.3.1 for the details). However, it needs bilinear groups the order of which is a product of two large primes, so it needs to deal with large group elements and the numbers of both attributes and possible values for each attribute specified in the ciphertext policy are fixed at the system setup while, in our constructions, the number of possible attribute values in the ciphertext policy can be increased and furthermore in our second construction, the number of

attributes in the ciphertext policy can be increased securely even after the system setup with the existing public parameters being unchanged.

Recently, Katz, Sahai, and Waters [KSW08] proposed a novel predicate (or *functional*) encryption scheme supporting *inner product* predicates and their scheme is very general and can realize both key-policy and ciphertext-policy ABE schemes. Their scheme can also realize hidden ciphertext policies that can be more expressive than ours. However, their scheme is based on a special type of bilinear groups the order of which is a product of three (or two) large primes while ours are not. Therefore, their scheme needs to deal with large group elements and requires new complexity assumptions for the security proof. By using the dual of the predicate corresponding to polynomial evaluation, the scheme in [KSW08] can realize the same access structure of ciphertext policies that our schemes can support (see Sect. 2.2.3.2 for the details) and then the ciphertext size of our schemes $O(\sum_{i=1}^{n} n_i)$ is comparable to that of [KSW08] where n is the number of attributes in ciphertext policies and n_i is the number of possible values for each attribute i. Fox example, if attribute i is boolean, $n_i = 2$. In the CP-ABE scheme of [KSW08], the maximum size of the subset of attribute values for each attribute specified in the ciphertext policy for successful decryption is fixed at the system setup while, in our constructions, the size can be increased. Also, the number of attributes specified in the ciphertext policy is fixed at the system setup while, in our second construction, the number of attributes in the ciphertext policy can be increased securely even after the system setup with the existing public parameters being unchanged. However, when the number of possible attribute values is huge, the scheme in [KSW08] is more advantageous than ours because it can enjoy the smaller ciphertext size and still realize the wildcard functionality.

Chase [Cha07] proposed a multi-authority ABE where multiple authorities generate secret keys for their monitored attributes. The technique of [Cha07] can be applicable to our schemes too. Abdalla et al. [ACD+06] proposed an identity-based encryption scheme where an encryptor can use wildcards to specify recipients of the ciphertext, but the positions of the wildcards and other ID components need to be revealed in the ciphertexts.

We summarize the comparison of major different schemes in Table 2.1.

2.2 Preliminaries

2.2.1 Bilinear Maps

We assume that there is an efficient algorithm ***Gen*** for generating bilinear groups. The algorithm ***Gen***, on input a security parameter 1^κ, outputs a tuple, $\boldsymbol{G} = [p, \mathbb{G}, \mathbb{G}_T, g \in \mathbb{G}, \boldsymbol{e}]$ where $\log_2(p) = \Theta(\kappa)$. A function $\boldsymbol{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a bilinear map. Here, $\mathbb{G}$ and $\mathbb{G}_T$ are multiplicative groups of prime order p, generated by g and $\boldsymbol{e}(g, g)$ respectively. The bilinear map $\boldsymbol{e}$ has the following properties:

1. Bilinearity: for all $a, b \in \mathbb{Z}_p$, $\boldsymbol{e}(g^a, g^b) = \boldsymbol{e}(g, g)^{ab}$.
2. Non-degeneracy: $\boldsymbol{e}(g, g) \neq 1$.

2.2.2 Complexity Assumptions

We describe complexity assumptions used in our security proofs.

2.2.2.1 The Decisional Bilinear Diffie-Hellman (DBDH) Assumption

We use the decisional version of the bilinear DH assumption [BF01, Jou00]. Let $z_1, z_2, z_3, z \in \mathbb{Z}_p^*$ be chosen at random and $g \in \mathbb{G}$ be a generator. The DBDH assumption is that no probabilistic polynomial-time algorithm can distinguish the tuple $[g, g^{z_1}, g^{z_2}, g^{z_3}, \boldsymbol{e}(g, g)^{z_1 z_2 z_3}]$ from the tuple $[g, g^{z_1}, g^{z_2}, g^{z_3}, \boldsymbol{e}(g, g)^{z}]$ with non-negligible advantage.

2.2.2.2 The Decision Linear (D-Linear) Assumption

The D-Linear assumption was first proposed in [BBS04]. Let $z_1, z_2, z_3, z_4, z \in \mathbb{Z}_p^*$ be chosen at random and $g \in \mathbb{G}$ be a generator. The D-Linear assumption is that no probabilistic polynomial-time algorithm can distinguish the tuple $[g, g^{z_1}, g^{z_2}, g^{z_1 z_3}, g^{z_2 z_4}, g^{z_3 + z_4}]$ from the tuple $[g, g^{z_1}, g^{z_2}, g^{z_1 z_3}, g^{z_2 z_4}, g^{z}]$ with non-negligible advantage.

2.2.3 Access Structure for Ciphertext

In the CP-ABE scheme, an encryptor specifies an access structure for a ciphertext, which is called a ciphertext policy. If a decryptor has a secret key whose associated set of attributes satisfies the access structure, she can decrypt the ciphertext. The access structures used in [BSW07] are the most flexible and expressive. For example, we can use an access structure such as ((A **AND** B) **OR** (C **AND** D)) in [BSW07]. This means that a recipient must have attributes A and B simultaneously or attributes C and D simultaneously in order to decrypt the ciphertext. Therefore, if a recipient has a secret key associated with a set of attributes {A, B, C}, she can satisfy the access structure and decrypt the ciphertext. However, if the recipient has a secret key associated with a set of attributes {A, C}, she can not satisfy the access structure or decrypt the ciphertext. Actually **AND**, **OR**, and threshold gates can be used for expressing the access structures in [BSW07].

However, the security proof of [BSW07] is in the *generic group model*. In order to obtain a reduction-based security proof, Cheung and Newport proposed another CP-ABE scheme [CN07] which is proved to be secure under *standard* complexity assumptions. The price of obtaining such security proofs is that the expressiveness of ciphertext policies in [CN07] is somewhat restricted as compared with [BSW07]. However, the expressiveness is not too restrictive and still remains useful.

The access structure and the attribute set associated with the secret key used in [CN07] are as follows. Let's assume that the total number of attributes in the system is n and the attributes are indexed as $\{\mathbb{A}_1, \mathbb{A}_2, \ldots, \mathbb{A}_i, \ldots, \mathbb{A}_n\}$ or we may use just i to refer to $\mathbb{A}_i$. We use the notation such as $L = [L_1, \ldots, L_n] = [1, 0, 1, \ldots, 0]$ in order to describe attribute-value pairs for a user, which we call the attribute list. For example, the user has the value 1 for $\mathbb{A}_1$, 0 for $\mathbb{A}_2$, 1 for $\mathbb{A}_3$, ..., and 0 for $\mathbb{A}_n$ in this case. A trusted authority generates a secret key for the user based on the user's attribute list.

In order to specify the access structure for a ciphertext, we use the notation such as $W = [W_1, \ldots, W_n] = [1, 1, *, *, 0]$ where $n = 5$, which we call the ciphertext policy. The wildcard $*$ can be used in the ciphtertext policies and it plays the role of "don't care" value. This can be considered as an **AND**-gate on all the attributes. For example, the above ciphertext policy means

that the recipient who wants to decrypt must have the value 1 for $\mathbb{A}_1$ and $\mathbb{A}_2$ and 0 for $\mathbb{A}_5$, and the values for $\mathbb{A}_3$ and $\mathbb{A}_4$ do not matter in the **AND**-gate. If the recipient has the secret key associated with, let us say, $[1, 1, 1, 0, 0]$, she can decrypt the ciphertext, but not if the secret key is associated with $[1, 1, 1, 0, 1]$.

Formally, given an attribute list $L = [L_1, L_2, \ldots, L_n]$ and a ciphertext policy $W = [W_1, W_2, \ldots, W_n]$, L satisfies W if, for all $i = 1, \ldots, n$, $L_i = W_i$ or $W_i = *$, and otherwise L does not satisfies W. We use the notation $L \models W$ to mean that L satisfies W.

In our constructions, we generalize the access structures in [CN07]. In [CN07], each attribute can take two values 1 and 0, but in our generalized access structures each attribute can take two or more values and each W_i in W can be any subset of possible values for $\mathbb{A}_i$. More formally, let $S_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,t}, \ldots, v_{i,n_i}\}$ be a set of possible values for $\mathbb{A}_i$ where n_i is the number of the possible values for $\mathbb{A}_i$. Then the attribute list L for a user is $L = [L_1, L_2, \ldots, L_i, \ldots, L_n]$ where $L_i \in S_i$ and the generalized ciphertext policy W is $W = [W_1, W_2, \ldots, W_i, \ldots, W_n]$ where $W_i \subseteq S_i$. The generalized ciphertext policy W means, let us say,

$$
\begin{aligned}
&(\mathbb{A}_1 = v_{1,1} \vee \mathbb{A}_1 = v_{1,3}) \\
&\quad \wedge (\mathbb{A}_2 = v_{2,2}) \wedge \ldots \\
&\quad \wedge (\mathbb{A}_i = v_{i,5} \vee \ldots \vee \mathbb{A}_i = v_{i,n_i}) \wedge \ldots \\
&\quad \wedge (\mathbb{A}_n = v_{n,1} \vee \mathbb{A}_n = v_{n,2} \vee \mathbb{A}_n = v_{n,3}).
\end{aligned}
$$

When the encryptor specifies a wildcard for $\mathbb{A}_i$, it corresponds to specifying $W_i = S_i$ for $\mathbb{A}_i$. The attribute list L satisfies the ciphertext policy W iff $L_i \in W_i$ for $1 \leq i \leq n$. We achieve recipient anonymity by hiding what subset W_i for each $\mathbb{A}_i$ is specified in the access structure of the **AND**-gate of all the attributes.

2.2.3.1 Realization of CP-ABE with [BW07]

We sketch how the scheme in [BW07] can realize the access structure of the ciphertext policy considered in this work by using HVE. For ease of exposition, suppose there are two attributes $\mathbb{A}_1, \mathbb{A}_2$ in the system and $\mathbb{A}_1$ can take values $v_{1,1}, v_{1,2}$ and $\mathbb{A}_2$ can take values $v_{2,1}, v_{2,2}, v_{2,3}$. When an encryptor encrypts a message, the encryptor specifies a

vector corresponding to $(v_{1,1}, v_{1,2}, v_{2,1}, v_{2,2}, v_{2,3})$ as a ciphertext policy. For example, if $(v_{1,1}, v_{1,2}, v_{2,1}, v_{2,2}, v_{2,3}) = (1, 0, 1, 0, 1)$, this means $(\mathbb{A}_1 = v_{1,1}) \wedge (\mathbb{A}_2 = v_{2,1} \vee \mathbb{A}_2 = v_{2,3})$. A decryptor with $\mathbb{A}_1 = v_{1,1} \wedge \mathbb{A}_2 = v_{2,3}$ obtains her secret key the vector of which corresponds to $(1, *, *, *, 1)$. The decryptor can decrypt the ciphertext if the vectors of both the ciphertext and the secret key match up except the wildcards. In this scheme, the length of the vectors (5 in the example above) is fixed at the system setup. Therefore, the numbers of both attributes and possible values for each attribute specified in the ciphertext policy are fixed at the system setup.

2.2.3.2 Realization of CP-ABE with [KSW08]

We sketch how the scheme in [KSW08] can realize the access structure of the ciphertext policy considered in this work by using the dual of the predicate corresponding to polynomial evaluation. Similarly, for ease of exposition, suppose there are two attributes $\mathbb{A}_1, \mathbb{A}_2$ in the system and $\mathbb{A}_1$ can take values $v_{1,1}, v_{1,2}$ and $\mathbb{A}_2$ can take values $v_{2,1}, v_{2,2}, v_{2,3}$. In this scheme, decryption succeeds if the vector for the ciphertext $(a_1, a_2, \ldots, a_n)$ and the vector for the secret key $(x_1, x_2, \ldots, x_n)$ satisfy the condition that $\sum_{i=1}^{n} a_i x_i = 0$.

When an encryptor encrypts a message with the ciphertext policy $(\mathbb{A}_1 = v_{1,1}) \wedge (\mathbb{A}_2 = v_{2,1} \vee \mathbb{A}_2 = v_{2,3})$, she prepares two polynomials $f_1(x) = c_1 x + c_0$ and $f_2(x) = d_2 x^2 + d_1 x + d_0$ such that $f_1(v_{1,1}) = 0$, $f_2(v_{2,1}) = 0$ and $f_2(v_{2,3}) = 0$ and specifies the vector $(c_1, c_0, d_2, d_1, d_0)$ as the ciphertext policy. A decryptor with $\mathbb{A}_1 = v_{1,1} \wedge \mathbb{A}_2 = v_{2,3}$ obtains her secret key the vector of which corresponds to $(v_{1,1}, 1, v_{2,3}^2, v_{2,3}, 1)$. For example, when the encryptor specifies a wildcard for attribute $\mathbb{A}_2$ in the ciphertext policy, she simply uses $f_2(x) = 0$ where $d_2 = d_1 = d_0 = 0$. In this scheme, the length of the vectors (5 in the example above) is fixed at the system setup. Therefore, the maximum size of the subset of attribute values for each attribute specified in the ciphertext policy for successful decryption is fixed at the system setup. Also, the number of attributes specified in the ciphertext policy is fixed at the system setup. However, when the number of possible attribute values is huge and the maximum size of the subset of attribute values specified in the ciphertext policy is small, the scheme in [KSW08] is more advantageous than ours because it can enjoy the smaller ciphertext size and still realize the wildcard functionality.

2.2.4 Syntax of CP-ABE

Our CP-ABE schemes consist of the following four algorithms.

Setup(1^κ). This algorithm takes the security parameter κ as input and generates a public key PK and a master secret key MK.

KeyGen(MK, L). This algorithm takes MK and an attribute list L as input and generates a secret key SK_L associated with L.

Encrypt(PK, M, W). This algorithm takes PK, a message M, and an ciphertext policy W as input, and generates a ciphertext CT.

Decrypt(CT, SK_L). This algorithm takes CT and SK_L associated with L as input and returns the message M if the attribute list L satisfies the ciphertext policy W specified for CT, that is, $L \models W$. If $L \not\models W$, it returns $\perp$ with overwhelming probability.

2.2.5 Security Model

We describe the security models for our CP-ABE. Based on [SBC+07, BW07, KSW08], we use the following security game. A CP-ABE scheme is selectively secure if no probabilistic polynomial-time adversary has non-negligible advantage in the following game.

Selective Game for CP-ABE

Init: The adversary commits to the challenge ciphertext policies W_0, W_1.

Setup: The challenger runs the ***Setup*** algorithm and gives PK to the adversary.

Phase 1: The adversary submits the attribute list L for a ***KeyGen*** query. If $(L \models W_0 \wedge L \models W_1)$ or $(L \not\models W_0 \wedge L \not\models W_1)$, the challenger gives the adversary the secret key SK_L. The adversary can repeat this polynomially many times.

Challenge: The adversary submits messages M_0, M_1 to the challenger. If the adversary obtained the SK_L whose associated attribute list L satisfies both W_0 and W_1 in Phase 1, then

it is required that $M_0 = M_1$. The challenger flips a random coin b and passes the ciphertext ***Encrypt***(PK, M_b, W_b) to the adversary.

Phase 2: Phase 1 is repeated. If $M_0 \neq M_1$, the adversary cannot submit L such that $L \models W_0 \wedge L \models W_1$.

Guess: The adversary outputs a guess b' of b.

The advantage of an adversary in this game is defined as $\left|\Pr[b' = b] - \frac{1}{2}\right|$ where the probability is taken over the random bits used by the challenger and the adversary. Since the adversary must commit to the challenge ciphertext policies before the setup phase, this model can be considered to be analogous to the selective-ID model [CHK03, CHK04] used in identity-based encryption schemes. In the non-selective-ID model, the adversary can specify the challenge ciphertext policies during the challenge phase. In the game, the adversary can submit L such that $L \models W_0$ and $L \models W_1$ if possible and then the adversary can decrypt the ciphertext. This definition captures that the adversary cannot obtain the useful information about the ciphertext policy more than the fact that she can decrypt the ciphertext. The above notion of security is called *match-concealing security* in [SBC+07].

2.3 Proposed Schemes

We construct two CP-ABE schemes that achieve recipient anonymity. In [CN07], the ciphertext policy needs to be revealed in the ciphertext so that a decryptor can know which secret key components should be used. Furthermore, in order to support wildcards for ciphertext policies, the public key components for the wildcards are prepared in [CN07] and the decryptor uses the secret key components corresponding to the wildcards if the wildcards are specified in the ciphertext policies. In our constructions, we can hide how the ciphertext policy is specified successfully.

First we show the construction of [CN07] and later explain the intuition behind our approach we take to make it recipient-anonymous. We assume, for notational simplicity, that the total number of attributes in the system is n and the attributes are indexed as $\{1, 2, \ldots, i, \ldots, n\}$.

2.3.1 Construction of [CN07]

The four algorithms are as follows:

Setup(1^κ). A trusted authority generates a tuple $\boldsymbol{G} = [p, \mathbb{G}, \mathbb{G}_T, g \in \mathbb{G}, \boldsymbol{e}] \leftarrow \boldsymbol{Gen}(1^\kappa)$, and random $w \in \mathbb{Z}_p^*$. For each attribute i where $1 \leq i \leq n$, the authority generates random values $a_i, \widehat{a_i}, a_i^*, \in \mathbb{Z}_p^*$. The authority computes $Y = \boldsymbol{e}(g,g)^w$ and $A_i = g^{a_i}, \widehat{A_i} = g^{\widehat{a_i}}, A_i^* = g^{a_i^*}$. The public key PK consists of $\langle Y, p, \mathbb{G}, \mathbb{G}_T, g, \boldsymbol{e}, \{A_i, \widehat{A_i}, A_i^*\}_{1 \leq i \leq n} \rangle$. The master secret key MK is $\langle w, \{a_i, \widehat{a_i}, a_i^*\}_{1 \leq i \leq n} \rangle$.

KeyGen(MK, L). Let $L = [L_1, L_2, \ldots, L_n]$ be the attribute list for the user who will obtain the corresponding secret key. The trusted authority picks up random values $s_i \in \mathbb{Z}_p^*$ for $1 \leq i \leq n$, sets $s = \sum_{i=1}^n s_i$, and computes $D_0 = g^{w-s}$. For $1 \leq i \leq n$, the authority also computes $[D_i, D_i^*] = [g^{s_i/a_i}, g^{s_i/a_i^*}]$ if $L_i = 1$, and $[D_i, D_i^*] = [g^{s_i/\widehat{a_i}}, g^{s_i/a_i^*}]$ if $L_i = 0$. The secret key SK_L is $\langle D_0, \{D_i, D_i^*\}_{1 \leq i \leq n} \rangle$.

Encrypt(PK, M, W). An encryptor encrypts a message $M \in \mathbb{G}_T$ under a ciphertext policy $W = [W_1, W_2, \ldots, W_n]$. The encryptor picks up a random value $r \in \mathbb{Z}_p^*$ and sets $\widetilde{C} = MY^r$ and $C_0 = g^r$. Also for $1 \leq i \leq n$, the encryptor computes C_i as follows: if $W_i = 1$, $C_i = A_i^r$; if $W_i = 0$, $C_i = \widehat{A_i}^r$; if $W_i = *$, $C_i = A_i^{*r}$. The ciphertext CT is $\langle \widetilde{C}, C_0, \{C_i\}_{1 \leq i \leq n} \rangle$. The encryptor needs to reveal W in CT so that recipients can know which secret key components should be used for each C_i.

Note that if W is hidden in CT, the recipients need to try all the possible combinations of the secret key components for decryption and it takes exponential time in n, which seems inefficient or impractical if we have a large number of attributes.

Decrypt(CT, SK_L). The recipient can check W to know whether $L \models W$. If $L \models W$, she can proceed. The recipient decrypts the CT, $\langle \widetilde{C}, C_0, \{C_i\}_{1 \leq i \leq n} \rangle$ by using her SK_L, $\langle D_0, \{D_i, D_i^*\}_{1 \leq i \leq n} \rangle$ associated with the attribute list L, as follows:

1. For $1 \leq i \leq n$,

$$D'_i = \begin{cases} D_i & \text{if } W_i \neq * \\ D^*_i & \text{if } W_i = *. \end{cases}$$

2.

$$M = \frac{\widetilde{C}}{\boldsymbol{e}(C_0, D_0) \prod_{i=1}^{n} \boldsymbol{e}(C_i, D'_i)}.$$

2.3.2 Main Idea

We describe how to make the construction of [CN07] recipient-anonymous. As mentioned earlier, we cannot have an efficient construction just by hiding W when the number of attributes n is large. To achieve our goal, the recipients need to be able to decrypt CT without knowing W and we also want to support wildcards in a hidden way. For that, we remove the public key components $\{A^*_i\}_{1 \leq i \leq n}$ for the wildcards and the secret key components $\{D^*_i\}_{1 \leq i \leq n}$ are not included in SK_L. Furthermore, instead of the ciphertext components $\{C_i\}_{1 \leq i \leq n}$, $\{C_i, \widehat{C_i}\}_{1 \leq i \leq n}$ are generated with $C_0 = g^r$ as follows: let $\{C_i, \widehat{C_i}\} = \{A_i^{r_1}, \widehat{A_i}^{r_2}\}$; if $W_i = 1$, we set $r_1 = r$ and r_2 is random; if $W_i = 0$, r_1 is random and $r_2 = r$; if $W_i = *$, $r_1 = r_2 = r$. That is, if $C_i = A_i^r$ or $\widehat{C_i} = \widehat{A_i^r}$, these ciphertext components are "*well-formed*" and can be used for successful decryption and otherwise "*malformed*" (or random). Each decryptor uses C_i for decryption if $L_i = 1$ and uses $\widehat{C_i}$ if $L_i = 0$ without knowing what is specified for W_i. By generating the ciphertext like this, we can realize the functionality of wildcards. We can generalize this idea to adapt to the multi-valued attribute setting.

Finally to make it hard to distinguish the well-formed components from the malformed components, we use the linear splitting technique in [BW06, SBC+07] and make our first construction provably secure as shown in Sect. 2.4.

2.3.3 Our First Construction

The four algorithms are as follows:

Setup(1^κ). A trusted authority generates a tuple $\boldsymbol{G} = [p, \mathbb{G}, \mathbb{G}_T, g \in \mathbb{G}, \boldsymbol{e}] \leftarrow$ ***Gen***(1^κ) and

random $w \in \mathbb{Z}_p^*$. For each attribute i where $1 \leq i \leq n$, the authority generates random values $\{a_{i,t}, b_{i,t} \in \mathbb{Z}_p^*\}_{1 \leq t \leq n_i}$ and random points $\{A_{i,t} \in \mathbb{G}\}_{1 \leq t \leq n_i}$ [*1]. The authority computes $Y = \boldsymbol{e}(g,g)^w$. The public key PK consists of $\langle Y, p, \mathbb{G}, \mathbb{G}_T, g, \boldsymbol{e}, \{\{A_{i,t}^{a_{i,t}}, A_{i,t}^{b_{i,t}}\}_{1 \leq t \leq n_i}\}_{1 \leq i \leq n}\rangle$. The master secret key MK is $\langle w, \{\{a_{i,t}, b_{i,t}\}_{1 \leq t \leq n_i}\}_{1 \leq i \leq n}\rangle$.

KeyGen(MK, L). Let $L = [L_1, L_2, \ldots, L_n] = [v_{1,t_1}, v_{2,t_2}, \ldots, v_{n,t_n}]$ be the attribute list for the user who obtains the corresponding secret key. The trusted authority picks up random values $s_i, \lambda_i \in \mathbb{Z}_p^*$ for $1 \leq i \leq n$, sets $s = \sum_{i=1}^n s_i$, and computes $D_0 = g^{w-s}$. For $1 \leq i \leq n$, the authority also computes $[D_{i,0}, D_{i,1}, D_{i,2}] = [g^{s_i}(A_{i,t_i})^{a_{i,t_i} b_{i,t_i} \lambda_i}, g^{a_{i,t_i}\lambda_i}, g^{b_{i,t_i}\lambda_i}]$ where $L_i = v_{i,t_i}$. The secret key SK_L is $\langle D_0, \{\{D_{i,j}\}_{0 \leq j \leq 2}\}_{1 \leq i \leq n}\rangle$.

Encrypt(PK, M, W). An encryptor encrypts a message $M \in \mathbb{G}_T$ under a ciphertext policy $W = [W_1, W_2, \ldots, W_n]$. The encryptor picks up a random value $r \in \mathbb{Z}_p^*$ and sets $\widetilde{C} = MY^r$ and $C_0 = g^r$. Also for $1 \leq i \leq n$, the encryptor picks up random values $\{r_{i,t} \in \mathbb{Z}_p^*\}_{1 \leq t \leq n_i}$ and computes $\{C_{i,t,1}, C_{i,t,2}\}_{1 \leq t \leq n_i}$ as follows: if $v_{i,t} \in W_i$, $[C_{i,t,1}, C_{i,t,2}] = [(A_{i,t}^{b_{i,t}})^{r_{i,t}}, (A_{i,t}^{a_{i,t}})^{r - r_{i,t}}]$ (*well-formed*); if $v_{i,t} \notin W_i$, $[C_{i,t,1}, C_{i,t,2}]$ are random (*malformed*). The ciphertext CT is $\langle \widetilde{C}, C_0, \{\{C_{i,t,1}, C_{i,t,2}\}_{1 \leq t \leq n_i}\}_{1 \leq i \leq n}\rangle$.

Decrypt(CT, SK_L). The recipient tries decrypting the CT, $\langle \widetilde{C}, C_0, \{\{C_{i,t,1}, C_{i,t,2}\}_{1 \leq t \leq n_i}\}_{1 \leq i \leq n}\rangle$ without knowing W by using her SK_L, $\langle D_0, \{\{D_{i,j}\}_{0 \leq j \leq 2}\}_{1 \leq i \leq n}\rangle$ associated with the attribute list L, as follows:

1. For $1 \leq i \leq n$,
$$[C'_{i,1}, C'_{i,2}] = [C_{i,t_i,1}, C_{i,t_i,2}] \text{ where } L_i = v_{i,t_i}.$$
2. $$M = \frac{\widetilde{C} \prod_{i=1}^n \boldsymbol{e}(C'_{i,1}, D_{i,1}) \boldsymbol{e}(C'_{i,2}, D_{i,2})}{\boldsymbol{e}(C_0, D_0) \prod_{i=1}^n \boldsymbol{e}(C_0, D_{i,0})}.$$

If the attribute list L satisfies the hidden ciphertext policy W of the CT, the recipient can decrypt the CT correctly. For the recipient to know whether the decryption was successful without knowing the ciphertext policy W, we can use the technique used in [BW07] in practice. As in

[*1] In the asymmetric bilinear groups, $A_{i,t}$ must be generated such that $A_{i,t} = g^{c_{i,t}}$ where $c_{i,t} \in_R \mathbb{Z}_p^*$ and $c_{i,t}$ is known to the authority so that the authority can use $c_{i,t}$ in ***KeyGen***.

[BW07], the encryptor picks up a random $k \in \mathbb{G}_T$ and derives two uniform and independent μ-bit symmetric keys (k_0, k_1) from k. The final ciphertext consists of $\langle k_1, \textbf{\textit{Encrypt}}(PK, k, W), E_{k_0}(M)\rangle$ where $\textbf{\textit{Encrypt}}(PK, k, W)$ is the ciphertext of k encrypted under PK and W, and $E_{k_0}(M)$ is the ciphertext of M encrypted under k_0 by using a symmetric encryption scheme. The recipient can use k_1 to check whether the decryption was successful after decrypting k where the false positive probability is approximately $1/2^{\mu}$. If successful, the recipient can decrypt M by using k_0 derived from k. The security proof is given in Sect. 2.4.

2.3.4 Second Construction with More Flexibility

We can also apply the technique in Sect. 2.3.2 to [BSW07] and make it recipient-anonymous. With a property inherited from [BSW07], this scheme is more flexible though the security proof is in the generic bilinear group model. The scheme in [BSW07] uses a symmetric bilinear group while we use an asymmetric bilinear group. That is, we assume $\textbf{\textit{Gen}}(1^{\kappa})$ outputs $\boldsymbol{G} = [p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2, \boldsymbol{e}]$ where $\boldsymbol{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear map. We also use the External Diffie-Hellman (XDH) assumption used in, for example, [BBS04, Sco02, CHL05] to achieve recipient anonymity, which holds on MNT curves [MNT01]. In the XDH assumption, it holds that the Decisional Diffie-Hellman (DDH) problem is hard in $\mathbb{G}_1$ and this implies that there does not exist an efficiently-computable isomorphism $\psi : \mathbb{G}_1 \rightarrow \mathbb{G}_2$. The four algorithms are as follows:

****Setup****(1^{κ}). A trusted authority generates a tuple $\boldsymbol{G} = [p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2, \boldsymbol{e}]$ and random $w, \beta \in \mathbb{Z}_p^*$. For each attribute i where $1 \leq i \leq n$, the authority generates random values $\{a_{i,t} \in \mathbb{Z}_p^*\}_{1 \leq t \leq n_i}$ and computes points $\{A_{i,t} = g_1^{a_{i,t}}\}_{1 \leq t \leq n_i}$. The authority computes $Y = \boldsymbol{e}(g_1, g_2)^w$ and $B = g_1^{\beta}$. The public key PK consists of $\langle Y, B, p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, \boldsymbol{e}, \{\{A_{i,t}\}_{1 \leq t \leq n_i}\}_{1 \leq i \leq n}\rangle$. The master secret key MK is $\langle w, \beta, \{\{a_{i,t}\}_{1 \leq t \leq n_i}\}_{1 \leq i \leq n}\rangle$.

****KeyGen****(MK, L). Let $L = [L_1, L_2, \ldots, L_n] = [v_{1,t_1}, v_{2,t_2}, \ldots, v_{n,t_n}]$ be the attribute list for the user who obtains the corresponding secret key. The trusted authority picks up random values $s, \lambda_i \in \mathbb{Z}_p^*$ for $1 \leq i \leq n$ and computes $D_0 = g_2^{\frac{w+s}{\beta}}$. For $1 \leq i \leq n$, the authority also computes $[D_{i,1}, D_{i,2}] = [g_2^{s + a_{i,t_i} \lambda_i}, g_2^{\lambda_i}]$ where $L_i = v_{i,t_i}$. The secret key SK_L is $\langle D_0, \{D_{i,1}, D_{i,2}\}_{1 \leq i \leq n}\rangle$.

****Encrypt****(PK, M, W). An encryptor encrypts a message $M \in \mathbb{G}_T$ under a ciphertext policy

$W = [W_1, W_2, \ldots, W_n]$. The encryptor picks up a random value $r \in \mathbb{Z}_p^*$ and sets $\widetilde{C} = MY^r$ and $C_0 = B^r$. Also for $1 \leq i \leq n$, the encryptor picks up random values $r_i \in \mathbb{Z}_p^*$ such that $r = \sum_{i=1}^{n} r_i$, sets $C_{i,1} = g_1^{r_i}$ and computes $\{C_{i,t,2}\}_{1 \leq t \leq n_i}$ as follows: if $v_{i,t} \in W_i$, $C_{i,t,2} = A_{i,t}^{r_i}$ (*well-formed*); if $v_{i,t} \notin W_i$, $C_{i,t,2}$ is random (*malformed*). The ciphertext CT is $\langle \widetilde{C}, C_0, \{C_{i,1}, \{C_{i,t,2}\}_{1 \leq t \leq n_i}\}_{1 \leq i \leq n} \rangle$.

Decrypt(CT, SK_L). The recipient decrypts the CT, $\langle \widetilde{C}, C_0, \{C_{i,1}, \{C_{i,t,2}\}_{1 \leq t \leq n_i}\}_{1 \leq i \leq n} \rangle$ by using her SK_L, $\langle D_0, \{D_{i,1}, D_{i,2}\}_{1 \leq i \leq n} \rangle$ associated with the attribute list L as follows:

1. For $1 \leq i \leq n$,
$$C'_{i,2} = C_{i,t_i,2} \text{ where } L_i = v_{i,t_i}.$$
2.
$$M = \frac{\widetilde{C} \prod_{i=1}^{n} \boldsymbol{e}(C_{i,1}, D_{i,1})}{\boldsymbol{e}(C_0, D_0) \prod_{i=1}^{n} \boldsymbol{e}(C'_{i,2}, D_{i,2})}.$$

Under the XDH assumption, it is hard to guess from CT what subset W_i the encryptor specified for each attribute $\mathbb{A}_i$ in the ciphertext policy. The security proof will be similar to that of [BSW07] and given in Sect. 2.6. We discuss the flexibility of this scheme in Sect. 2.7 in more detail.

2.4 Overview of Security Proofs for First Construction

We prove that our first scheme is selectively secure under the DBDH assumption and the D-Linear assumption. We will give the high-level arguments of the proofs here and the detailed proofs of the lemmas are given in Sect. 2.5.

Suppose the adversary commits to the challenge ciphertext policies W_0, W_1 at the beginning of the game. We use the notation $W_b = [W_{b,1}, W_{b,2}, \ldots, W_{b,i}, \ldots, W_{b,n}]$.

The proof uses a sequence of hybrid games to argue that the adversary cannot win the original security game denoted by G with non-negligible probability. We begin by slightly modifying the game G into a game G_0. Games G and G_0 are the same except for how the challenge ciphertext is generated. In G_0, if $M_0 \neq M_1$, then the challenge ciphtertext component $\widetilde{C}$ is a random element of $\mathbb{G}_T$ regardless of the random coin b. The rest of the ciphertext is generated as usual. If $M_0 = M_1$,

then the challenge ciphertext in G_0 is generated correctly. That is, $G = G_0$ in this case.

Lemma 1 *Under the DBDH assumption, for any polynomial time adversary $\mathcal{A}$, the difference of advantage of $\mathcal{A}$ in game G and game G_0 is negligible in the security parameter κ.*

Next, we modify G_0 by changing how to generate the ciphertext components $\{\{C_{i,t,1}, C_{i,t,2}\}_{1\le t\le n_i}\}_{1\le i\le n}$ and define a sequence of games as follows. Note that, in G_ℓ where $\ell > 0$, the challenge ciphtertext component $\widetilde{C}$ is generated as in G_0.

For $v_{i,t}$ such that $(v_{i,t} \in W_{0,i} \wedge v_{i,t} \in W_{1,i})$ or $(v_{i,t} \notin W_{0,i} \wedge v_{i,t} \notin W_{1,i})$, the components $\{C_{i,t,1}, C_{i,t,2}\}$ are generated as in the real scheme through the sequence of all the games.

If there is $v_{i,t}$ such that $(v_{i,t} \in W_{0,i} \wedge v_{i,t} \notin W_{1,i})$ or $(v_{i,t} \notin W_{0,i} \wedge v_{i,t} \in W_{1,i})$, the components $\{C_{i,t,1}, C_{i,t,2}\}$ generated properly in game $G_{\ell-1}$ are replaced with the random values in the new modified game G_ℓ regardless of the random coin b. Every time we replace such components $\{C_{i,t,1}, C_{i,t,2}\}$ with the random values, we define a new modified game. We repeat this replacement one by one until we have no component that satisfies $(v_{i,t} \in W_{0,i} \wedge v_{i,t} \notin W_{1,i})$ or $(v_{i,t} \notin W_{0,i} \wedge v_{i,t} \in W_{1,i})$. In the last game of the sequence, the advantage of the adversary is zero because the adversary is given a ciphertext chosen from the same distribution regardless of the random coin b.

By replacing the well-formed ciphertext components in $G_{\ell-1}$ with the random values in G_ℓ in this way, we can embed a D-Linear challenge in the ciphertext such that the distinguisher of $G_{\ell-1}$ and G_ℓ leads to the distinguisher of the D-Linear challenge.

Lemma 2 *Under the D-Linear assumption, for any polynomial time adversary $\mathcal{A}$, the difference of advantage of $\mathcal{A}$ in game $G_{\ell-1}$ and game G_ℓ is negligible in the security parameter κ.*

By considering the sequence $G, G_0, G_1, \ldots$ of games starting with the original game G, no polynomial adversary can win the game G with non-negligible advantage by the lemmas above.

2.5 Proofs of Lemmas

2.5.1 Proof of Lemma 1

Proof: We prove our lemma by assuming that a polynomial adversary $\mathcal{A}$ has non-negligible difference ϵ between its advantage in game G and its advantage in game G_0. We build a simulator $\mathcal{B}$ that can play the DBDH game with advantage ϵ.

Given a DBDH challenge $[g, g^{z_1}, g^{z_2}, g^{z_3}, Z]$ by the challenger where Z is either $\boldsymbol{e}(g,g)^{z_1 z_2 z_3}$ or random with equal probability, the simulator $\mathcal{B}$ creates the following simulation.

Init: The simulator $\mathcal{B}$ runs $\mathcal{A}$. $\mathcal{A}$ gives $\mathcal{B}$ two challenge chiphertext policies $W_0 = [W_{0,1}, \ldots, W_{0,n}], W_1 = [W_{1,1}, \ldots, W_{1,n}]$. Then $\mathcal{B}$ flips a random coin $b \in \{0, 1\}$.

Setup: To provide a public key PK to $\mathcal{A}$, $\mathcal{B}$ sets Y to $\boldsymbol{e}(g,g)^{z_1 z_2}$. This implies $w = z_1 z_2$. For each attribute i where $1 \leq i \leq n$, $\mathcal{B}$ generates $\{A_{i,t}\}_{1 \leq t \leq n_i}$ such that $A_{i,t} = g^{\alpha_{i,t}}$ if $v_{i,t} \in W_{b,i}$ and $A_{i,t} = g^{z_1 \alpha_{i,t}}$ if $v_{i,t} \notin W_{b,i}$ where $\{\alpha_{i,t} \in \mathbb{Z}_p^*\}_{1 \leq t \leq n_i}$ are random. Then $\mathcal{B}$ publishes public parameters as in the real scheme by picking up $\{a_{i,t}, b_{i,t}\}_{1 \leq t \leq n_i}$ at random for $1 \leq i \leq n$.

Phase 1: $\mathcal{A}$ submits an attribute list $L = [L_1, \ldots, L_n]$ in a secret key query. We consider only the case where $L \not\models W_0 \wedge L \not\models W_1$. The reason for this is by our definition if $L \models W_0 \wedge L \models W_1$, then the challenge messages M_0, M_1 will be equal. In this case, the games G and G_0 are the same, so there can be no difference of advantage of $\mathcal{A}$ in G and G_0. Therefore, $\mathcal{B}$ simply aborts and takes a random guess.

When $L \not\models W_0 \wedge L \not\models W_1$, there must be $k \in \{1, \ldots, n\}$ such that $L_k (= v_{k,t_k}) \notin W_{b,k}$.

For $1 \leq i \leq n$, $\mathcal{B}$ picks up $s_i' \in \mathbb{Z}_p^*$ at random. It then sets $s_k = z_1 z_2 + s_k'$ and for every $i \neq k$, sets $s_i = s_i'$. Finally it sets $s = \sum_{i=1}^n s_i = z_1 z_2 + \sum_{i=1}^n s_i'$. The D_0 component of the secret key can be computed as

$$D_0 = g^{w-s} = g^{z_1 z_2 - s} = g^{-\sum_{i=1}^n s_i'}.$$

For k, $\mathcal{B}$ computes the components $[D_{k,0}, D_{k,1}, D_{k,2}] = [g^{s_k} (A_{k,t_k})^{a_{k,t_k} b_{k,t_k} \lambda_k}, g^{a_{k,t_k} \lambda_k}, g^{b_{k,t_k} \lambda_k}]$ as

follows:

$$\begin{aligned} D_{k,0} &= g^{s_k}(A_{k,t_k})^{a_{k,t_k} b_{k,t_k} \lambda_k} \\ &= g^{z_1 z_2 + s'_k}(A_{k,t_k})^{a_{k,t_k} b_{k,t_k} \lambda_k} \\ &= g^{z_1 z_2 + s'_k}(g^{z_1 \alpha_{k,t_k}})^{a_{k,t_k} b_{k,t_k} \lambda_k} \\ &= g^{s'_k}(g^{z_1 \alpha_{k,t_k}})^{a_{k,t_k} b_{k,t_k} \lambda'_k} \end{aligned}$$

where λ_k is chosen at random such that

$$\lambda_k = -\frac{z_2}{\alpha_{k,t_k} a_{k,t_k} b_{k,t_k}} + \lambda'_k$$

and random λ'_k is known to $\mathcal{B}$.

$\mathcal{B}$ can compute the components $[D_{k,1}, D_{k,2}]$ easily.

For $i \neq k$, $\mathcal{B}$ can also compute $[D_{i,0}, D_{i,1}, D_{i,2}]$ easily.

Challenge: $\mathcal{A}$ submits two challenge messages M_0 and M_1. If $M_0 = M_1$, $\mathcal{B}$ simply aborts and takes a random guess for the reason given above. Otherwise $\mathcal{B}$ sets $\widetilde{C} = M_b Z$ and $C_0 = g^{z_3}$ which implies $r = z_3$ and generates, for W_b, the ciphertext $\langle \widetilde{C}, C_0, \{\{C_{i,t,1}, C_{i,t,2}\}_{1 \leq t \leq n_i}\}_{1 \leq i \leq n} \rangle$. When $v_{i,t} \in W_{b,i}$, $\mathcal{B}$ can generate $\{C_{i,t,1}, C_{i,t,2}\}$ correctly because $A_{i,t}$ does not contain unknown z_1 and when $v_{i,t} \notin W_{b,i}$, $\{C_{i,t,1}, C_{i,t,2}\}$ can be simply chosen at random.

Phase 2: Phase 1 is repeated.

Guess: $\mathcal{A}$ outputs a guess b' of b. If $b' = b$, $\mathcal{B}$ outputs 1 and otherwise outputs 0. By our assumption, the probability that $\mathcal{A}$ guesses b correctly in game G has a non-negligible ϵ difference from that of it guessing b correctly in G_0. When $Z = \boldsymbol{e}(g,g)^{z_1 z_2 z_3}$, $\mathcal{A}$ is in game G and when Z is random, $\mathcal{A}$ is in game G_0. Therefore the simulator $\mathcal{B}$ has advantage ϵ in the DBDH game.

□

2.5.2 Proof of Lemma 2

Proof: We prove our lemma by assuming that a polynomial adversary $\mathcal{A}$ has non-negligible difference ϵ between its advantage in game $G_{\ell-1}$ and its advantage in game G_ℓ. We build a simulator $\mathcal{B}$ that can play the D-Linear game with advantage ϵ.

Given a D-Linear challenge $[g, g^{z_1}, g^{z_2}, Z, g^{z_2 z_4}, g^{z_3+z_4}]$ by the challenger where Z is either $g^{z_1 z_3}$ or random with equal probability, the simulator $\mathcal{B}$ creates the simulation. Note that this D-Linear assumption is equivalent to that of Sect. 2.2.2.2.

As mentioned in Sect. 2.4, in $G_{\ell-1}$, the ciphertext components $\{C_{i_\ell,t_\ell,1}, C_{i_\ell,t_\ell,2}\}$ are generated as in the real scheme, whereas, in G_ℓ, the components are random regardless of the random coin b and we assume that $(v_{i_\ell,t_\ell} \in W_{1,i_\ell} \wedge v_{i_\ell,t_\ell} \notin W_{0,i_\ell})$ without loss of generality.

Init: The simulator $\mathcal{B}$ runs $\mathcal{A}$. $\mathcal{A}$ gives $\mathcal{B}$ two challenge chiphertext policies $W_0 = [W_{0,1}, \ldots, W_{0,n}], W_1 = [W_{1,1}, \ldots, W_{1,n}]$. Then $\mathcal{B}$ flips a random coin $b \in \{0,1\}$. If $b = 0$, $\mathcal{B}$ aborts and takes a random guess. The reason for this is by our definition if $b = 0$ where $(v_{i_\ell,t_\ell} \in W_{1,i} \wedge v_{i_\ell,t_\ell} \notin W_{0,i})$, we have $G_{\ell-1} = G_\ell$ because the distribution of the challenge ciphertext in game $G_{\ell-1}$ is the same as that of game G_ℓ, so there can be no difference of advantage of $\mathcal{A}$ in $G_{\ell-1}$ and G_ℓ. We proceeds assuming $b = 1$.

Setup: To provide a public key PK to $\mathcal{A}$, $\mathcal{B}$ sets Y to $\boldsymbol{e}(g,g)^w$ where w is known to $\mathcal{B}$. For each attribute i where $1 \le i \le n$, $\mathcal{B}$ generates $\{A_{i,t}\}_{1\le t\le n_i}$ such that $A_{i,t} = g^{\alpha_{i,t}}$ if $v_{i,t} \in W_{b,i}$ and $A_{i,t} = g^{z_1\alpha_{i,t}}$ if $v_{i,t} \notin W_{b,i}$ where $\{\alpha_{i,t} \in \mathbb{Z}_p^*\}_{1\le t\le n_i}$ are random. Then $\mathcal{B}$ publishes public parameters as in the real scheme by picking up $\{a_{i,t}, b_{i,t}\}_{1\le t\le n_i}$ at random for $1 \le i \le n$ with the exception that, for a_{i_ℓ,t_ℓ} and b_{i_ℓ,t_ℓ}, $\mathcal{B}$ sets $a_{i_\ell,t_\ell} = z_1$ and $b_{i_\ell,t_\ell} = z_2$ and can compute $A_{i_\ell,t_\ell}^{a_{i_\ell,t_\ell}} = g^{\alpha_{i_\ell,t_\ell} a_{i_\ell,t_\ell}}$ and $A_{i_\ell,t_\ell}^{b_{i_\ell,t_\ell}} = g^{\alpha_{i_\ell,t_\ell} b_{i_\ell,t_\ell}}$ without knowing z_1, z_2.

Phase 1: $\mathcal{A}$ submits an attribute list $L = [L_1, \ldots, L_n]$ in a secret key query. If $L_{i_\ell} \neq v_{i_\ell,t_\ell}$, $\mathcal{B}$ can generate the corresponding secret key easily.

Let's assume $L_{i_\ell} = v_{i_\ell,t_\ell}$. $\mathcal{B}$ needs to compute the secret key components $[D_{i_\ell,0}, D_{i_\ell,1}, D_{i_\ell,2}] = [g^{s_{i_\ell}} (A_{i_\ell,t_\ell})^{a_{i_\ell,t_\ell} b_{i_\ell,t_\ell} \lambda_{i_\ell}}, g^{a_{i_\ell,t_\ell}\lambda_{i_\ell}}, g^{b_{i_\ell,t_\ell}\lambda_{i_\ell}}]$ where $a_{i_\ell,t_\ell} = z_1, b_{i_\ell,t_\ell} = z_2$.

$\mathcal{B}$ can compute $D_{i_\ell,0}$ as

$$\begin{aligned} D_{i_\ell,0} &= g^{s_{i_\ell}} (A_{i_\ell,t_\ell})^{a_{i_\ell,t_\ell} b_{i_\ell,t_\ell} \lambda_{i_\ell}} \\ &= g^{s_{i_\ell}} (A_{i_\ell,t_\ell})^{z_1 z_2 \lambda_{i_\ell}} \\ &= g^{s_{i_\ell}} (g^{\alpha_{i_\ell,t_\ell}})^{z_1 z_2 \lambda_{i_\ell}} \\ &= g^{s'_{i_\ell}} \end{aligned}$$

where s_{i_ℓ} is chosen at random such that

$$s_{i_\ell} = s'_{i_\ell} - \alpha_{i_\ell,t_\ell} z_1 z_2 \lambda_{i_\ell}$$

and random s'_{i_ℓ} is known to $\mathcal{B}$. $\mathcal{B}$ can compute the components $[D_{i_\ell,1}, D_{i_\ell,2}]$ easily without knowing z_1, z_2.

Here we can assume $L \not\models W_0 \wedge L \not\models W_1$ because $L_{i_\ell} = v_{i_\ell,t_\ell} \wedge v_{i_\ell,t_\ell} \notin W_{1-b,i_\ell}$. That is, we have $L \not\models W_{1-b}$ and therefore $L \not\models W_b$, so there must be $k \in \{1, \ldots, n\}$ such that $L_k (= v_{k,t_k}) \notin W_{b,k}$. Then $\mathcal{B}$ generates $[D_{k,0}, D_{k,1}, D_{k,2}]$ as follows: $\mathcal{B}$ sets $s_k = s'_k + \alpha_{i_\ell,t_\ell} z_1 z_2 \lambda_{i_\ell}$ where s'_k is random and known to $\mathcal{B}$ and computes

$$\begin{aligned} D_{k,0} &= g^{s_k} (A_{k,t_k})^{a_{k,t_k} b_{k,t_k} \lambda_k} \\ &= g^{s'_k + \alpha_{i_\ell,t_\ell} z_1 z_2 \lambda_{i_\ell}} (g^{z_1 \alpha_{k,t_k}})^{a_{k,t_k} b_{k,t_k} \lambda_k} \\ &= g^{s'_k} (g^{z_1 \alpha_{k,t_k}})^{a_{k,t_k} b_{k,t_k} \lambda'_k} \end{aligned}$$

where λ_k is chosen at random such that

$$\lambda_k = \lambda'_k - \frac{\alpha_{i_\ell,t_\ell} z_2 \lambda_{i_\ell}}{\alpha_{k,t_k} a_{k,t_k} b_{k,t_k}}$$

and random λ'_k is known to $\mathcal{B}$. $\mathcal{B}$ can compute the components $[D_{k,1}, D_{k,2}]$ easily without knowing z_2.

Also, for $i \neq i_\ell, k$, $\mathcal{B}$ can compute $[D_{i,0}, D_{i,1}, D_{i,2}]$ easily.

Finally by computing

$$\begin{aligned} s &= \sum_{i=1}^{n} s_i \\ &= s_{i_\ell} + s_k + \sum_{i \neq i_\ell, k} s_i \\ &= s'_{i_\ell} - \alpha_{i_\ell,t_\ell} z_1 z_2 \lambda_{i_\ell} + s'_k + \alpha_{i_\ell,t_\ell} z_1 z_2 \lambda_{i_\ell} + \sum_{i \neq i_\ell, k} s_i \\ &= s'_{i_\ell} + s'_k + \sum_{i \neq i_\ell, k} s_i, \end{aligned}$$

the component $D_0 = g^{w-s}$ of the secret key can be computed.

Challenge: $\mathcal{A}$ submits two challenge messages M_0 and M_1. $\mathcal{B}$ sets $C_0 = g^{z_3+z_4}$ which implies $r = z_3 + z_4$. If $M_0 \neq M_1$, $\mathcal{B}$ sets $\widetilde{C}$ to be random and if $M_0 = M_1$, $\mathcal{B}$ sets $\widetilde{C} = M_b \boldsymbol{e}(g, g^{z_3+z_4})^w$. $\mathcal{B}$ generates, for W_b, the ciphertext components $\{\{C_{i,t,1}, C_{i,t,2}\}_{1 \leq t \leq n_i}\}_{1 \leq i \leq n}$ as in $G_{\ell-1}$ with the exception that the components $\{C_{i_\ell,t_\ell,1}, C_{i_\ell,t_\ell,2}\}$ are computed as

$$C_{i_\ell,t_\ell,1} = (A_{i_\ell,t_\ell}^{b_{i_\ell,t_\ell}})^{r_{i_\ell,t_\ell}} = (A_{i_\ell,t_\ell}^{z_2})^{z_4} = (g^{\alpha_{i_\ell,t_\ell} z_2})^{z_4},$$

$$C_{i_\ell,t_\ell,2} = (A_{i_\ell}^{a_{i_\ell,t_\ell}})^{r - r_{i_\ell,t_\ell}} = (g^{\alpha_{i_\ell,t_\ell} z_1})^{z_3} = Z^{\alpha_{i_\ell,t_\ell}}$$

without knowing $z_2 z_4, z_1 z_3$. This implies that $r_{i_\ell,t_\ell} = z_4$ and $Z = g^{z_1 z_3}$ and if $Z = g^{z_1 z_3}$, the components are well-formed and $\mathcal{A}$ is in game $G_{\ell-1}$.

Phase 2: Phase 1 is repeated.

Guess: $\mathcal{A}$ outputs a guess b' of b. If $b' = b$, $\mathcal{B}$ outputs 1 and otherwise outputs 0. By our assumption, the probability that $\mathcal{A}$ guesses b correctly in game $G_{\ell-1}$ has a non-negligible ϵ difference from that of it guessing b correctly in G_ℓ. When $Z = g^{z_1 z_3}$, $\mathcal{A}$ is in game $G_{\ell-1}$ and when Z is random, $\mathcal{A}$ is in game G_ℓ. Therefore the simulator $\mathcal{B}$ has advantage ϵ in the D-Linear game.

□

2.6 Security Proof for Second Construction

Generic Bilinear Group Model. The generic group model was introduced in [Nec94, Sho97] and extended to the bilinear group setting in [BB04, BBG05]. Under this model, elements of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ of prime order p are assumed to be encoded as unique random strings so that only equality may be tested between the group elements by the adversary. Let $\xi_1 : \mathbb{Z}_p \to \{0,1\}^*$ be the random encoding of elements of $\mathbb{G}_1$, i.e., an injective map and then $\mathbb{G}_1 = \{\xi_1(x) : x \in \mathbb{Z}_p\}$. If $g_1 \in \mathbb{G}_1$ is a random generator of $\mathbb{G}_1$, $\xi_1(x)$ can be considered to be the random string representation of $g_1^x \in \mathbb{G}_1$. Similarly we define ξ_2, ξ_T for $\mathbb{G}_2, \mathbb{G}_T$. In order to perform the group and pairing operations, the adversary needs to interact with an oracle that performs the group and pairing operations using those random strings. That is, the adversary communicates with the operation oracle using only the ξ-representations of the group elements. In this model, the adversary can make the following oracle queries.

Multiplication: Given $\xi_k(a), \xi_k(b)$ by the adversary, the oracle returns $\xi_k(a+b)$ where $k \in \{1, 2, T\}$.

Exponentiation by a constant: Given $\xi_k(a)$ and $c(\in \mathbb{Z}_p)$ by the adversary, the oracle returns $\xi_k(ca)$ where $k \in \{1, 2, T\}$ and c is a constant known to the adversary.

Pairing: Given $\xi_1(a), \xi_2(b)$ by the adversary, the oracle returns $\xi_T(ab)$ that corresponds to $\boldsymbol{e}(\xi_1(a), \xi_2(b))$.

Homomorphism: Given $\xi_2(a)$ by the adversary, the oracle returns $\xi_1(a)$. Inverse homomorphism queries are impossible because of the XDH assumption.

As in [BBG05, BSW07], we prove the following theorem.

Theorem 1 *Let $\xi_1, \xi_2, \xi_T, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be defined as above. For any adversary $\mathcal{A}$, let q be a bound on the total number of group elements it receives from queries it makes to the operation oracle and from its interaction with the non-selective-ID CP-ABE game. Then we have that the advantage of the adversary in the game is $O\left(\frac{q^2}{p}\right)$.*

Proof: We can create a simulator $\mathcal{B}$ that interacts with $\mathcal{A}$ as follows:

$\mathcal{B}$ maintains three lists: $L_{\mathbb{G}_1} = \{(f_{1,\ell}, \xi_{1,\ell}) : \ell = 1, \ldots, \tau_1\}$, $L_{\mathbb{G}_2} = \{(f_{2,\ell}, \xi_{2,\ell}) : \ell = 1, \ldots, \tau_2\}$, $L_{\mathbb{G}_T} = \{(f_{T,\ell}, \xi_{T,\ell}) : \ell = 1, \ldots, \tau_T\}$. The items $f_{1,\ell}$, $f_{2,\ell}$, $f_{T,\ell}$ contain rational functions or constants. $\mathcal{B}$ uses $f_{1,\ell}$, $f_{2,\ell}$, $f_{T,\ell}$ to store the group operation queries that $\mathcal{A}$ makes and $\xi_{1,\ell}, \xi_{2,\ell}, \xi_{T,\ell}$ to store the query results. That is, $\xi_{1,\ell} = \xi_1(f_{1,\ell}), \xi_{2,\ell} = \xi_2(f_{2,\ell})$, and $\xi_{T,\ell} = \xi_T(f_{T,\ell})$.

At the beginning of the CP-ABE game, $\mathcal{B}$ sets $f_{1,1} = 1$, $f_{2,1} = 1$, $f_{T,1} = 1$ and chooses their corresponding random strings $\xi_{1,1}, \xi_{2,1}, \xi_{T,1}$. That is, $\xi_{1,1}, \xi_{2,1}$, and $\xi_{T,1}$ correspond to g_1, g_2, and $\boldsymbol{e}(g_1, g_2)$ respectively. Also $\mathcal{B}$ updates the lists by adding the tuples corresponding to public key components $\boldsymbol{e}(g_1, g_2)^w, g_1^\beta$, and $\{\{g_1^{a_{i,t}}\}_{1 \le t \le n_i}\}_{1 \le i \le n}$. Note that, in the real non-selective-ID CP-ABE game, the challenger chooses random real values for the variables $\langle w, \beta, \{\{a_{i,t}\}_{1 \le t \le n_i}\}_{1 \le i \le n} \rangle$ and maintains them in the lists. However, in the simulation $\mathcal{B}$ does not choose real values for the variables and instead maintains multi-variate rational functions in the lists. At the end of the simulation, $\mathcal{B}$ chooses each random value for each variable using lazy evaluation and reveals all the tuples in the lists so that $\mathcal{A}$ can verify the consistency of the game. To start the game, $\mathcal{B}$ sends to $\mathcal{A}$ the chosen random strings in the lists. Whenever $\mathcal{A}$ makes the oracle queries, secret key queries, and challenge query, $\mathcal{B}$ updates its lists as follows. Note that whenever $\mathcal{B}$ adds new tuples to the lists, the new random strings are returned to $\mathcal{A}$.

Multiplication: $\mathcal{A}$ inputs $\xi_k(a)$ and $\xi_k(b)$ where $k \in \{1, 2, T\}$. $\mathcal{B}$ checks that $\xi_k(a)$ and $\xi_k(b)$ are in the list $L_{\mathbb{G}_k}$ and returns $\perp$ if they are not. Then $\mathcal{B}$ computes $f = a + b \bmod p$. If f is already in the list $L_{\mathbb{G}_k}$, then $\mathcal{B}$ returns $\xi_k(f)$. Otherwise, $\mathcal{B}$ chooses a new random string $\xi_k(f)$ and adds a new tuple $(f, \xi_k(f))$ to the list.

Exponentiation by a constant: $\mathcal{A}$ inputs $\xi_k(a)$ and a constant c where $k \in \{1, 2, T\}$. $\mathcal{B}$ checks that $\xi_k(a)$ is in the list $L_{\mathbb{G}_k}$ and returns $\perp$ if it is not. Then $\mathcal{B}$ computes $f = ca \bmod p$. If f is already in the list $L_{\mathbb{G}_k}$, then $\mathcal{B}$ returns $\xi_k(f)$. Otherwise, $\mathcal{B}$ chooses a new random string $\xi_k(f)$ and adds a new tuple $(f, \xi_k(f))$ to the list.

Pairing: $\mathcal{A}$ inputs $\xi_1(a)$ and $\xi_2(b)$. $\mathcal{B}$ checks that $\xi_1(a)$ and $\xi_2(b)$ are in the lists $L_{\mathbb{G}_1}$ and $L_{\mathbb{G}_2}$ respectively and returns $\perp$ if they are not. Then $\mathcal{B}$ computes $f = ab \bmod p$. If f is already in the list $L_{\mathbb{G}_T}$, then $\mathcal{B}$ returns $\xi_T(f)$. Otherwise, $\mathcal{B}$ chooses a new random string $\xi_T(f)$ and adds a new tuple $(f, \xi_T(f))$ to the list.

Homomorphism: $\mathcal{A}$ inputs $\xi_2(a)$. $\mathcal{B}$ checks that $\xi_2(a)$ is in the list $L_{\mathbb{G}_2}$ and returns $\perp$ if it is not.

If a is already in the list $L_{\mathbb{G}_1}$, then $\mathcal{B}$ returns $\xi_1(a)$. Otherwise, $\mathcal{B}$ chooses a new random string $\xi_1(a)$ and adds a new tuple $(a, \xi_1(a))$ to the list.

Secret Key Query: $\mathcal{A}$ inputs an attribute list $L = [L_1, L_2, \ldots, L_n] = [v_{1,t_1}, v_{2,t_2}, \ldots, v_{n,t_n}]$ for the j-th query. $\mathcal{B}$ adds to the list $L_{\mathbb{G}_2}$ new tuples corresponding to SK_L $\langle g_2^{\frac{w+s^{(j)}}{\beta}}, \{g_2^{s^{(j)}+a_{i,t_i}\lambda_i^{(j)}}, g_2^{\lambda_i^{(j)}}\}_{1\le i\le n}\rangle$ that include new variables $s^{(j)}$ and $\lambda_i^{(j)}$'s.

Challenge Query: $\mathcal{A}$ inputs $\langle M_0, W_0\rangle$ and $\langle M_1, W_1\rangle$ where $W_i = [W_{i,1}, \ldots, W_{i,n}]$ and $i \in \{0,1\}$. We assume that $M_0 = \xi_T(m_0)$ and $M_1 = \xi_T(m_1)$ already exist in the list $L_{\mathbb{G}_T}$. In the real game, the challenger chooses random $b \in \{0,1\}$ to encrypt M_b for W_b. However, $\mathcal{B}$ creates the ciphertext $\langle \widetilde{C}, C_0, \{C_{i,1}, \{C_{i,t,2}\}_{1\le t\le n_i}\}_{1\le i\le n}\rangle$ as follows:

For C_0, $\mathcal{B}$ adds a tuple $(\beta r, \xi_1(\beta r))$ to the list $L_{\mathbb{G}_1}$ where r is a new variable. For $\{C_{i,1}\}_{1\le i\le n}$, $\mathcal{B}$ adds tuples $(r_i, \xi_1(r_i))$ to the list $L_{\mathbb{G}_1}$ where r_i's are new variables and $r = \sum_{i=1}^{n} r_i$.

If $M_0 = M_1$, $\mathcal{B}$ adds to the list $L_{\mathbb{G}_T}$ a tuple $(m_0 + wr, \xi_T(m_0 + wr))$ for $\widetilde{C}$. Note that $m_0 = m_1$ in this case. If $M_0 \neq M_1$, $\mathcal{B}$ adds to the list $L_{\mathbb{G}_T}$ a tuple $(\theta_{\widetilde{C}}, \xi_T(\theta_{\widetilde{C}}))$ for $\widetilde{C}$ where $\theta_{\widetilde{C}}$ is a new variable.

For $\{\{C_{i,t,2}\}_{1\le t\le n_i}\}_{1\le i\le n}$, if $v_{i,t} \in W_{0,i} \wedge v_{i,t} \in W_{1,i}$, $\mathcal{B}$ adds a tuple $(a_{i,t}r_i, \xi_1(a_{i,t}r_i))$ to the list $L_{\mathbb{G}_1}$. If $v_{i,t} \notin W_{0,i} \wedge v_{i,t} \notin W_{1,i}$, $\mathcal{B}$ adds a tuple $(r_{i,t}, \xi_1(r_{i,t}))$ to the list $L_{\mathbb{G}_1}$ where $r_{i,t}$'s are new variables. If $(v_{i,t} \notin W_{0,i} \wedge v_{i,t} \in W_{1,i})$ or $(v_{i,t} \in W_{0,i} \wedge v_{i,t} \notin W_{1,i})$, $\mathcal{B}$ adds a tuple $(\theta_{a_{i,t}r_i}, \xi_1(\theta_{a_{i,t}r_i}))$ to the list $L_{\mathbb{G}_1}$ where $\theta_{a_{i,t}r_i}$'s are new variables.

After $\mathcal{A}$ terminates and returns a guess $b' \in \{0,1\}$ of b, $\mathcal{B}$ chooses random $b \in \{0,1\}$ and subsitutes $m_b + wr$ for $\theta_{\widetilde{C}}$ if $\theta_{\widetilde{C}}$ was used in $L_{\mathbb{G}_T}$ and substitutes $a_{i,t}r_i$ for $\theta_{a_{i,t}r_i}$ where $v_{i,t} \in W_{b,i} \wedge v_{i,t} \notin W_{1-b,i}$. Finally, $\mathcal{B}$ chooses random values from $\mathbb{Z}_p$ for all the variables and reveals all the evaluated tuples in the lists.

Analysis of $\mathcal{B}$'s Simulation. The $\mathcal{B}$'s simulation is perfect if substituting the chosen random values for all the variables does not create any equality relation (i.e., unexpected collision) among intermediate rational functions that is not an equality of rational functions. If no unexpected collisions occur, the success probability of $\mathcal{A}$ is $\frac{1}{2}$ because the $\mathcal{B}$'s simulation is perfect. Such unexpected collisions occur only when

$$f_{k,\ell} = f_{k,\ell'} \text{ in } \mathbb{Z}_p \text{ for some } \ell, \ell', \text{ yet } f_{k,\ell} \neq f_{k,\ell'} \text{ as rational functions where } k \in \{1, 2, T\}.$$

The probability that such unexpected collisions occur in $L_{\mathbb{G}_1}, L_{\mathbb{G}_2}$ or $L_{\mathbb{G}_T}$ is at most $O\left(\frac{q^2}{p}\right)$ by the Schwartz-Zippel lemma [Zip79, Sch80] as in [BBG05, BSW07].

Next we show that no new equalities between rational functions such as $f_{k,\ell} = f_{k,\ell'}$ are created even if $\mathcal{B}$ substitutes variables $m_b + wr$ and $a_{i,t}r_i$'s for variables such as $\theta_{\widetilde{C}}$ and $\theta_{a_{i,t}r_i}$'s at the end of the simulation as described earlier. That is, we show that it cannot happen that $f_{k,\ell} \neq f_{k,\ell'}$ before $\mathcal{B}$'s variable substitution but $f_{k,\ell} = f_{k,\ell'}$ after $\mathcal{B}$'s variable substitution. This means that we must show that $\mathcal{A}$ cannot construct a query for $f(= f_{k,\ell} - f_{k,\ell'})$ where $f \neq 0$ before $\mathcal{B}$'s variable substitution and $f = 0$ after $\mathcal{B}$'s variable substitution. As in [BSW07], we will show that $\mathcal{A}$ can never construct queries for $\boldsymbol{e}(g_1, g_2)^{\gamma wr}$ or $\boldsymbol{e}(g_1, g_2)^{\gamma' a_{i,t}r_i}$'s by an exhaustive case analysis where γ and γ' are any terms.

Case of wr: When $\mathcal{B}$ substitutes $m_b + wr$ for $\theta_{\widetilde{C}}$, it means that $\mathcal{A}$ cannot obtain SK_L such that $L \models W_0 \wedge L \models W_1$ because $M_0 \neq M_1$. Therefore, $\mathcal{A}$ cannot decrypt the ciphertext even if $\mathcal{B}$ substitutes $a_{i,t}r_i$'s for $\theta_{a_{i,t}r_i}$'s. As proved in [BSW07], in this case, $\mathcal{A}$ cannot construct a query for $\boldsymbol{e}(g_1, g_2)^{\gamma wr}$.

Case of $a_{i,t}r_i$: Fix any $a_{i,t}r_i$ that appears when $\mathcal{B}$ applies variable substitution at the end of the simulation. Obviously, $\mathcal{A}$ cannot construct a query for $g_1^{\gamma' a_{i,t}r_i}$, so if $g_1^{a_{i,t}r_i}$ is testable (i.e., $\mathcal{A}$ can test whether $\theta_{a_{i,t}r_i} = a_{i,t}r_i$), $\mathcal{A}$ must be able to construct a query for $\boldsymbol{e}(g_1, g_2)^{\gamma' a_{i,t}r_i}$. If $\mathcal{A}$ can do so, $\mathcal{A}$ can test whether $\theta_{a_{i,t}r_i} = a_{i,t}r_i$ by comparing $\xi_T(\gamma' a_{i,t}r_i)$ with $\xi_T(\gamma' \theta_{a_{i,t}r_i})$. In other words, if $g_1^{a_{i,t}r_i}$ is testable, $\mathcal{A}$ must be able to construct a query for $\boldsymbol{e}(g_1, g_2)^{\nu}$ where ν is a non-zero rational function including the variable $\theta_{a_{i,t}r_i}$ and becomes zero (or vanishes) when $a_{i,t}r_i$ is substituted for $\theta_{a_{i,t}r_i}$ and other variables are also substituted appropriately in ν. Note that if it can happen that $f_{T,\ell} \neq f_{T,\ell'}$ before $\mathcal{B}$'s variable substitution and $f_{T,\ell} = f_{T,\ell'}$ after $\mathcal{B}$'s variable substitution, $\mathcal{A}$ can construct such a query for $\boldsymbol{e}(g_1, g_2)^{\nu}$ where $\nu = f_{T,\ell} - f_{T,\ell'}$. Therefore, we show that $\mathcal{A}$ can never construct such a query for $\boldsymbol{e}(g_1, g_2)^{\nu}$ and no new equalities between rational functions in $L_{\mathbb{G}_T}$ are created after $\mathcal{B}$ applies variable substitution at the end of the simulation.

Suppose that $\mathcal{A}$ could construct such ν. To cancel $a_{i,t}r_i$ in ν that appears after $\mathcal{B}$ substitutes $a_{i,t}r_i$ for $\theta_{a_{i,t}r_i}$, $\mathcal{A}$ needs to pair $g_1^{r_i}(= \xi_1(r_i))$ with $g_2^{s^{(j)} + a_{i,t}\lambda_i^{(j)}}(= \xi_2(s^{(j)} + a_{i,t}\lambda_i^{(j)}))$ or to pair $g_1^{a_{i,t'}r_i}(= \xi_1(a_{i,t'}r_i))$ with $g_2^{s^{(j)} + a_{i,t}\lambda_i^{(j)}}(= \xi_2(s^{(j)} + a_{i,t}\lambda_i^{(j)}))$ to create a term in ν that includes $a_{i,t}r_i$

where $t \neq t'$.

If $\mathcal{A}$ pairs $g_1^{a_{i,t'} r_i} (= \xi_1(a_{i,t'} r_i))$ with $g_2^{s^{(j)} + a_{i,t} \lambda_i^{(j)}} (= \xi_2(s^{(j)} + a_{i,t} \lambda_i^{(j)}))$, $\mathcal{A}$ needs to pair $g_1^{\theta_{a_{i,t} r_i}} (= \xi_1(\theta_{a_{i,t} r_i}))$ with $g_2^{a_{i,t'} \lambda_i^{(j)}} (= \xi_2(a_{i,t'} \lambda_i^{(j)}))$ to make ν vanish, but this is impossible because $g_2^{a_{i,t'} \lambda_i^{(j)}} (= \xi_2(a_{i,t'} \lambda_i^{(j)}))$ is not available.

If $\mathcal{A}$ pairs $g_1^{r_i} (= \xi_1(r_i))$ with $g_2^{s^{(j)} + a_{i,t} \lambda_i^{(j)}} (= \xi_2(s^{(j)} + a_{i,t} \lambda_i^{(j)}))$, $\mathcal{A}$ needs to cancel $\boldsymbol{e}(g_1, g_2)^{s^{(j)} r_i} (= \xi_T(s^{(j)} r_i))$. In this case, $\mathcal{A}$ might be able to obtain $\boldsymbol{e}(g_1, g_2)^{wr} (= \xi_T(wr))$ and $\boldsymbol{e}(g_1, g_2)^{s^{(j)} r} (= \xi_T(s^{(j)} r))$ because $\mathcal{A}$ can make a secret key query for SK_L such that $L \models W_0 \wedge L \models W_1$. On the other hand, we know that $v_{i,t} \in W_{b,i} \wedge v_{i,t} \notin W_{1-b,i}$. This means that $\mathcal{A}$ cannot have SK_L such that $L \models W_0 \wedge L \models W_1$ and $L_i = v_{i,t}$. That is, $\mathcal{A}$ cannot have SK_L such that SK_L can decrypt the ciphertext and $L_i = v_{i,t}$. By the argument similar to that of [BSW07], there is $i' (\neq i)$ such that $\mathcal{A}$ cannot cancel $\boldsymbol{e}(g_1, g_2)^{r_{i'} a_{i',t'} \lambda_{i'}^{(j)}} (= \xi_T(r_{i'} a_{i',t'} \lambda_{i'}^{(j)}))$ even if $\mathcal{A}$ obtains $\boldsymbol{e}(g_1, g_2)^{wr} (= \xi_T(wr))$ and $\boldsymbol{e}(g_1, g_2)^{s^{(j)} r} (= \xi_T(s^{(j)} r))$. Thus $\mathcal{A}$ cannot cancel $\boldsymbol{e}(g_1, g_2)^{s^{(j)} r_i} (= \xi_T(s^{(j)} r_i))$ and cannot construct a query for a non-zero rational function ν that becomes zero when $a_{i,t} r_i$ is substituted for $\theta_{a_{i,t} r_i}$.

Therefore, no new equalities between rational functions $f_{k,\ell} = f_{k,\ell'}$ in the lists are created where $k \in \{1, 2, T\}$ even if $\mathcal{B}$ substitutes variables $m_b + wr$ and $a_{i,t} r_i$'s for variables such as $\theta_{\widetilde{C}}$ and $\theta_{a_{i,t} r_i}$'s at the end of the simulation. Thus the probability that the unexpected collisions occur is still at most $O\left(\frac{q^2}{p}\right)$. This concludes the proof. □

2.7 Adding Attributes after ***Setup***

In our schemes, it is easy to add new possible values $v_{i,t}$'s of each attribute $\mathbb{A}_i$ in the ciphertext policy even after ***Setup*** is executed, because we have only to add the public key components for the new values of $\mathbb{A}_i$ and the existing public parameters can remain unchanged. That is, the access structure for the ciphertext policy can be extended accordingly though the ciphertext size is also increased. However, in our first scheme, it cannot be done securely to simply add new attributes $\mathbb{A}_{i'}$'s in the ciphertext policy with the existing public parameters being unchanged after ***Setup*** is executed and some users already have their secret keys. The reason is as follows. Suppose there are three attributes $\mathbb{A}_1, \mathbb{A}_2, \mathbb{A}_3$ in the system when ***Setup*** is executed and a user obtains her

secret key SK_L where the attribute list $L = [L_1, L_2, L_3] = [1, 1, 0]$. After that, a new attribute $\mathbb{A}_4$ is added in the system and the corresponding public key components for $\mathbb{A}_4$ are generated and published. Then an encryptor may specify a ciphertext policy $W = [W_1, \ldots, W_4] = [*, *, 0, 1]$, requiring the legitimate recipients to have the value 1 for $\mathbb{A}_4$. In this case, the user who has the above SK_L can decrypt the ciphertext encrypted under the ciphertext policy W even if she does not have the secret key component for $\mathbb{A}_4$, because L satisfies $[W_1, W_2, W_3]$ partially and it enables the user to combine all the secret key components to reconstruct $s = \sum_{i=1}^{n} s_i$ in the exponent for decryption. The similar situations can also happen in [CN07, KSW08, BW07, SBC+07] if we consider the setting where new attributes may be added in the ciphertext policy dynamically after ***Setup*** is executed. As mentioned in [OSW07], we may be able to prepare redundant filler attributes reserved for future use, but it increases the ciphertext size unnecessarily.

The second scheme can avoid this situation with the property inherited from [BSW07] and we can add new attributes in the ciphertext policy securely after ***Setup*** is executed where the existing public parameters can remain unchanged. Note that in this scheme, the encryptor splits random r in the ciphertext CT such that $r = \sum_{i=1}^{n} r_i$ and it forces decryptors to have the secret key components for all the attributes specified in the ciphertext policy even if the attributes in the ciphertext policy were added after the decryptors obtained their secret keys. If a user wants to decrypt the ciphertext with the ciphertext policy including newly added attributes, she must obtain a new secret key including the newly added attributes from the trusted authority again.

Additionally, in the second scheme, an encryptor can specify a variable-length ciphertext policy. For example, the encryptor can specify the ciphertext policy $W = [W_{i_1}, W_{i_2}, \ldots, W_{i_m}]$ where $m < n$ and n is the number of all the attributes in the system. Since there are several attributes that do not appear in the ciphertext policy, the partial information on the ciphertext policy is leaked. That is, it means that the wildcards are specified for the attributes not appearing in the ciphertext policy. However, it may be acceptable to the encryptor in some cases and it can reduce the size of the ciphertext.

Table. 2.1 Comparison of different schemes

	Expressiveness of policy	Anonymity	Complexity assumption	Type of bilinear group	Add attrs after setup
[BW07]	**AND**-gates on multi-valued attributes with wildcards	yes	cDBDH, C3DH	group of composite order $N = pq$	no
[BSW07]	all boolean formula	no	generic group model	any	yes
[CN07]	**AND**-gates on postive and negative attributes with wildcards	no	DBDH	any	no
[KSW08]	all boolean formula	yes	new assumptions based on composite order group	group of composite order $N = pqr$	no
This work1	**AND**-gates on multi-valued attributes with wildcards	yes	DBDH, D-Linear	any	no
This work2	**AND**-gates on multi-valued attributes with wildcards	yes	generic group model	DDH-hard group	yes

Chapter 3

Multiparty Computation for Integer Arithmetic Primitives

Damgård *et al.* [DFK+06] showed a novel technique to convert a polynomial sharing of secret a into the sharings of the bits of a in constant rounds, which is called the bit-decomposition protocol. The bit-decomposition protocol is a very powerful tool because it enables bit-oriented operations even if shared secrets are given as elements in the field. However, the bit-decomposition protocol is relatively expensive.

In this chapter, we present a simplified bit-decomposition protocol by analyzing the original protocol. Moreover, we construct more efficient protocols for a comparison, interval test and equality test of shared secrets without relying on the bit-decomposition protocol though it seems essential to such bit-oriented operations. The key idea is that we do computation on secret a with c and r where $c = a + r$, c is a revealed value, and r is a random bitwise-shared secret. The outputs of these protocols are also shared without being revealed.

The realized protocols as well as the original protocol are constant-round and run with less communication rounds and less data communication than those of [DFK+06]. For example, the round complexities are reduced by a factor of approximately 3 to 10. The results in this chapter were published as [NO07].

3.1 Introduction

3.1.1 Background

Secure *multiparty computation* (MPC) allows a set of mutually distrustful parties to jointly compute an agreed function of their inputs in such a way that the correctness of the output and the privacy of the parties' inputs are guaranteed. That is, when a function is represented as $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$, each party with its private input x_i obtains only the output y_i but nothing else.

A great deal of work (e.g., [Yao82, GMW87, BGW88, CCD88, BMR90, FKN94, JJ00]) has been done in this research field. By using generic circuit based protocols, it is shown that any function can be computed securely [BGW88, GMW87]. However, the general protocols tend to be inefficient; hence the main aim of our research is to construct efficient protocols for specific functions.

When we are interested in integer arithmetic, there are two choices to represent a function: an arithmetic circuit over a prime field $\mathbb{Z}_p$ and a Boolean circuit. Inputs (and outputs) in the arithmetic circuit are represented as elements in $\mathbb{Z}_p$ (or a ring), while inputs in the Boolean circuit are represented as bits. The input encoding has an influence on the efficiency of computation. Addition and multiplication of shared secrets can be performed efficiently in the arithmetic circuit, whereas not in the Boolean circuit. On the other hand, bit-oriented operations like interval tests, equality tests, and comparisons of shared secrets are easy in the Boolean circuit, whereas they are non-trivial tasks in the arithmetic circuit.

To bridge the gap between arithmetic circuits and Boolean circuits, Damgård *et al.* [DFK+06] have proposed the MPC protocol called bit-decomposition in the secret sharing setting (e.g., [BGW88, GRR98]). Also, Schoenmakers and Tuyls [ST06] have proposed a similar protocol for MPC [CDN01, DN03] based on threshold homomorphic cryptosystems (THC) [DJ01, FPS00]. In the bit-decomposition protocol, a sharing of an element in the field (or an encryption of an element in the ring in the threshold homomorphic setting) is converted into sharings (encryptions) of bits.

The bit-decomposition protocol is very useful and has many applications because it enables bit-oriented operations to be performed in the arithmetic circuit without performing the entire computation bitwise. For example, when computing a^b by using the techniques in [ACS02, DFK+06], or the Hamming distance between a and b where shared secrets a and b are elements in $\mathbb{Z}_p$, the bit-decomposition protocol is essential because we need the bitwise sharings of the shared secrets. Other important applications are comparisons, interval tests and equality tests of shared secrets. For example, in the comparison protocol, a single shared bit is computed such that it indicates the result of a comparison between two shared secrets. In the Boolean circuit, it is relatively easy to compare two shared secrets because the bits of the secrets are shared. That is, in the comparison protocol based on the Boolean circuit (which we call the bitwise less-than protocol in Sect. 3.3.5 as in [DFK+06]), we can check the secrets *bit by bit* privately and compare the two shared secrets even without revealing the bit position that determines the comparison result. Therefore, even if inputs are given as sharings of elements in the field, the comparison can be performed easily with the bit-decomposition protocol.

Thus the bit-decomposition protocol is a very powerful tool because changing the representations of shared secrets enables us to gain the benefits of both Boolean circuits and arithmetic circuits. However, the bit-decomposition protocol involves expensive computation in terms of round and communication complexities.

In this dissertation, we present a simplified bit-decomposition protocol by analyzing the original protocol. Moreover, we construct more efficient protocols for the main applications of the bit-decomposition protocol, which are interval tests, equality tests, and comparisons, without relying on the bit-decomposition protocol though it seemed essential. For example, the equality test protocol is an important subprotocol in [CD01, OK05, MF06], so it will be meaningful to construct efficient protocols for these applications without relying on the bit-decomposition protocol if possible. For the equality test, we present deterministic and probabilistic protocols.

In our constructions, the outputs of the protocols are also shared without being revealed, so they can be secret inputs for the subsequent computation. Therefore, our protocols can be used as building blocks in the more complex computation.

3.1.2 Our Contributions

We construct constant-round protocols for bit-decomposition, interval test, comparison, and equality test, building on the subprotocols in [DFK+06]. The proposed bit-decomposition protocol runs with less communication rounds and less data communication than the original protocol [DFK+06]. Therefore, the interval test, comparison and equality test protocols are also improved inevitably by using the proposed bit-decomposition protocol. However, we present new protocols dedicated to them without relying on the bit-decomposition protocol. By using our protocols, given shared secrets as elements in $\mathbb{Z}_p$, we can perform the interval tests, equality tests, and comparisons of the shared secrets more efficiently than the bit-decomposition based protocols. For the equality test, we propose two kinds of protocols. One (Proposed1) is a deterministic protocol and the other (Proposed2) is a probabilistic protocol with a negligible error probability and a much smaller round complexity. The key idea is that we do computation on secret a with c and r where $c = a + r$, c is a revealed value, and r is a random bitwise-shared secret.

In Table 3.1, we summarize the results of the round and communication (comm.) complexities of each protocol where ℓ is the bit length of prime p of the underlying field for linear secret sharing schemes and k must be chosen such that the error probability $\left(\frac{1}{2}\right)^k$ is negligible. Here "BD-based" means that the protocol is based on the proposed bit-decomposition protocol. As shown in Table 3.1, we can see that these bit-oritented operations can be realized with smaller complexities than those of the bit-decomposition based protocols by constructing them without the bit-decomposition protocol. For example, the round complexities are reduced by a factor of approximately 3 to 10.

Our protocols (except the probabilistic equality test protocol which is only applicable to the secret sharing setting) are applicable to both the secret sharing setting [DFK+06] and the threshold homomorphic setting [ST06] though we describe our constructions based on the secret sharing setting.

Table. 3.1 Comparison of Round / Communication Complexities

Protocol		Round	Comm.
Bit-Decomposition	[DFK+06]	38	$93\ell + 94\ell \log_2 \ell$
	Proposed	25	$93\ell + 47\ell \log_2 \ell$
Interval Test	[DFK+06]	44	$127\ell + 94\ell \log_2 \ell + 1$
	BD-based	31	$127\ell + 47\ell \log_2 \ell + 1$
	Proposed	13	$110\ell + 1$
Comparison	[DFK+06]	44	$205\ell + 188\ell \log_2 \ell$
	BD-based	31	$205\ell + 94\ell \log_2 \ell$
	Proposed	15	$279\ell + 5$
Equality Test	[DFK+06]	39	$98\ell + 94\ell \log_2 \ell$
	BD-based	26	$98\ell + 47\ell \log_2 \ell$
	Proposed1	8	81ℓ
	Proposed2	4	$10k$

3.1.3 Related Work

Damgård *et al.* [DFK+06] have shown a novel technique to convert a polynomial sharing of an element in $\mathbb{Z}_p$ into sharings of bits in constant rounds. Also Shoenmakers and Tuyls [ST06] have shown a similar conversion technique for multiparty computation based on threshold homomorphic cryptosystems [CDN01, DN03]. These protocols are the first to bridge the gap between arithmetic circuits and Boolean circuits.

Toft [Tof07] has proposed another version of a probabilistic equality test protocol independently of and concurrently with our probabilistic equality test protocol. Both the protocols use the property of quadratic residues in a probabilistic way.

Recently, as a practical approach (rather than theoretical constant-round protocols), in [BDJ+06, FJ06, Tof05], the implementation for multiparty integer computation, including the bit-decomposition and comparison, is described with non-constant-round protocols where shared

secrets are assumed to be sufficiently small compared with prime p of the underlying secret sharing scheme, whereas we do not assume that shared secrets are upper bounded by a certain value as in [DFK+06]. We mention this aspect in Sect. 3.7.

3.2 Preliminaries

We assume that n parties $P_1, \ldots, P_n$ are mutually connected by secure and authenticated channels in a synchronous network and the index i for each P_i is public among the parties. Let p be an odd prime and ℓ be the bit length of p. $\mathbb{Z}_p$ is a prime field. When we write $a \in \mathbb{Z}_p$, it means that $a \in \{0, 1, \ldots, p-1\}$. We use $[a]_{\mathbb{F}_p}$ to denote a polynomial sharing [Sha79] (see Sect. 3.3.1.1) of secret $a \in \mathbb{Z}_p$ which is equal to a finite field $\mathbb{F}_p$. We also use the simplified notation $[a]_p$ instead of $[a]_{\mathbb{F}_p}$ if the context is clear. The polynomial sharing $[a]_p$ means that a is shared among the parties where f_a is a random polynomial $f_a(x) = a + a_1 x + a_2 x^2 + \cdots + a_t x^t \bmod p$ with randomly chosen $a_i \in \mathbb{Z}_p$ for $1 \leq i \leq t$, $t < \frac{n}{2}$, and $f_a(i)$ is the P_i's share of a. An adversary can corrupt up to t parties. We describe our protocols in the so-called "honest-but-curious" model, but standard techniques will be applicable to make our protocols robust.

Let C be a Boolean test. When we write $[C]_p$, it means that $C \in \{0, 1\}$ and $C = 1$ iff C is true. For example, we use $[a < b]_p$ to denote the output of the comparison protocol.

Because the multiplication protocol is a dominant factor of the complexity, as in [DFK+06], we measure the round complexity of a protocol by the number of rounds of parallel invocations of the multiplication protocol [GRR98] and we also measure the communication complexity by the number of invocations of the multiplication protocol. The round complexity relates to the time required for a protocol to be completed and the communication complexity relates to the amount of data communicated among the parties during a protocol run. Though our measurement of complexities basically follows that of [DFK+06], the complexity analysis in [DFK+06] is rough. In this dissertation, we reevaluate the round and communication complexities of the protocols in [DFK+06] to compare our protocols with those of [DFK+06] based on the same measurement.

3.3 Building Blocks

3.3.1 Core Primitives

3.3.1.1 Shamir Secret Sharing

The concept of secret sharing was proposed in [Sha79]. By using this scheme, a secret is shared among multiple parties, and can be reconstructed by collecting a certain number of secret shares.

Now we describle how to construct a $(t+1, n)$ threshold secret sharing scheme. In order to share a secret s, the dealer of s generates a polynomial f with a_i chosen at random from $\{0, 1, \ldots, p-1\}$,

$$f(x) = s + a_1 x + a_2 x^2 + \cdots + a_t x^t \bmod p$$

where $0 \leq s < p$ and p is a prime. The parties are numbered from 1 to n where $n < p$ is assumed. The dealer sends the secret share $f(i)$ to the i-th party P_i through the secure channel. Any subset S of $t+1$ parties out of n total parties can reconstruct $s = f(0)$ by using Lagrange interpolation where

$$f(x) = \sum_{i \in S} f(i) \lambda_{x,i}^S \bmod p$$

and

$$\lambda_{x,i}^S = \prod_{i' \in S \setminus \{i\}} \frac{(x - i')}{(i - i')}.$$

The value $\lambda_{x,i}^S$ is a Lagrange coefficient. The important thing to note is that the adversary can obtain no partial information even if he collects t secret shares. This can be confirmed as follows. Suppose the adversary has collected t secret shares of a subset S'. Then for each possible guess s_{guess}, a valid polynomial $r(x)$ exists such that,

$$r(x) = \left(\sum_{i \in S'} f(i) \lambda_{x,i}^{S' \cup \{0\}} \right) + \left(s_{\text{guess}} \lambda_{x,0}^{S' \cup \{0\}} \right) \bmod p.$$

Therefore, the adversary cannot guess the correct secret. This scheme can also be used over any extension field $\mathbb{F}_{p^e}$.

3.3.1.2 Homomorphic Cryptosystem with Threshold Decryption

We sketch the concept of threshold homomorphic cryptosystems (THCs) for completeness. An additively homomorphic public key cryptosystem (e.g., [Pai99]) has the following properties.

- $E(m_1 + m_2) = E(m_1) \times E(m_2)$
- $E(cm) = E(m)^c$
- $E(-m) = E(m)^{-1}$

where $E(m)$ is the ciphertext of m and c is a public value. The above computation can be done without knowing the plaintexts m_1, m_2, m or the decryption key.

In MPC based on THCs, the decryption key is shared among the parties by using secret sharing as in [DJ01, FPS00] and the threshold decryption can be done only if the threshold number of parties cooperate and the decryption key itself is never revealed even after the decryption is done correctly. With these properties, a shared secret in the secret sharing setting corresponds to an encryption of the secret and the reconstruction of the secret by Lagrange interpolation corresponds to the threshold decryption and therefore we can realize the same MPC functionality in both the secret sharing and threshold homomorphic settings.

3.3.2 Distributed Computation with Shared Secrets for Addition and Multiplication

We utilize the classical BGW protocol [BGW88] based on Shamir secret sharing [Sha79]. Let's assume now that n parties have two shared secrets a and b as $[a]_p = \{f_a(1), \ldots, f_a(n)\}$ and $[b]_p = \{f_b(1), \ldots, f_b(n)\}$. Then the parties can obtain $[c + a \bmod p]_p, [ca \bmod p]_p$, and $[a + b \bmod p]_p$ easily where c is a public constant as follows: To compute $[c + a \bmod p]_p, [ca \bmod p]_p$, and $[a + b \bmod p]_p$, each P_i has only to locally compute $c + f_a(i) \bmod p, c f_a(i) \bmod p$, and $f_a(i) + f_b(i) \bmod p$ respectively. Therefore, these can be done efficiently without communication among n parties. When we write $[c + a]_p = c + [a]_p, [ca]_p = c[a]_p$, and $[a + b]_p = [a]_p + [b]_p$, these mean that the parties perform these operations. We also use $\sum$, for example, like $\sum_{i=1}^{3} [a_i]_p$ to denote $[a_1]_p + [a_2]_p + [a_3]_p$.

Multiplication to obtain $[ab \bmod p]_p$ is a bit more complex and it requires the parties to communicate with each other (see Sect. 3.3.3 for the details). When we write $[ab]_p = [a]_p \times [b]_p$, it means that the parties perform the multiplication protocol to compute $[ab \bmod p]_p$.

3.3.3 Multiplication Protocol

3.3.3.1 In Secret Sharing Setting

The multiplication protocol in [BGW88] was very complex and a simple and efficient multiplication protocol was proposed in [GRR98]. We sketch the multiplication protocol in [GRR98].

Let's assume that the parties have two t-degree polynomial sharings $[a]_p = \{f_a(1), \ldots, f_a(n)\}$ and $[b]_p = \{f_b(1), \ldots, f_b(n)\}$ such that

$$f_a(x) = a + a_1 x + \cdots + a_t x^t \bmod p$$

and

$$f_b(x) = b + b_1 x + \cdots + b_t x^t \bmod p$$

and they want to compute a t-degree polynomial sharing $[ab \bmod p]_p$. We consider a $2t$-degree polynomial $f_{ab}(x) = f_a(x) f_b(x)$. Then we can notice that $f_{ab}(0) = ab$ and $f_{ab}(i) = f_a(i) f_b(i)$. Therefore, $2t + 1 (\leq n)$ shares can reconstruct ab. That is, by using Lagrange interpolation, ab can be reconstructed with the Lagrange coefficients $\lambda^S_{0,i}$ as follows.

$$ab = f_{ab}(0) = \sum_{i \in S} \lambda^S_{0,i} f_{ab}(i),$$

$$\lambda^S_{x,i} = \prod_{i' \in S \setminus \{i\}} \frac{x - i'}{i - i'}$$

where S can be any subset such that $S \subseteq \{1, 2, \ldots, n\}$ and $|S| = 2t + 1$. Note that the Lagrange coefficients can be computed by anyone because they involve no secret information.

Because P_i can compute $f_{ab}(i) = f_a(i) f_b(i)$ locally, for $i \in S$, we let P_i reshare $f_{ab}(i)$ by a t-degree polynomial sharing $[f_{ab}(i)]_p$. Finally, the parties can compute $[ab]_p$ by

$$[ab]_p = \sum_{i \in S} \lambda^S_{0,i} [f_{ab}(i)]_p.$$

3.3.3.2 In Threshold Homomorphic Setting

We also sketch the multiplication protocol in [DN03] for completeness.

Now let's assume that there are two ciphertexts $E(a)$ and $E(b)$ and that parties want to compute $E(ab)$. Let N be the plaintext domain for a homomorphic cryptosystem. First each party P_i picks up $r\_i \in \mathbb{Z}_N$ at random and broadcasts $E(r\_i)$ and $E(r\_ib)$. Let $r = \sum_{i=1}^{n} r\_i$. All parties can now compute $E(r) = E(\sum_{i=1}^{n} r\_i)$ and $E(rb) = E(\sum_{i=1}^{n} r\_ib)$. Next, the parties cooperate to decrypt $E(a + r)$ by threshold decryption without revealing the decryption key itself. Then all parties can compute $E(b * (a + r)) = E(b)^{a+r}$ because $a + r$ is public. The final result $E(ab)$ can be computed as

$$E(ab) = E(b * (a + r)) * E(br)^{-1} = E(b * (a + r)) * E(-br).$$

3.3.3.3 Round and Communication Complexities

We explain the difference between the round and communication complexities with a concrete example. Let's assume that the parties compute $[a]_p \times [b]_p \times [c]_p \times [d]_p$. If the parties perform the multiplication protocol sequentially, the parties obtain $[ab]_p, [abc]_p$, and $[abcd]_p$ in order. Then the round complexity of this computation is 3 rounds and the communication complexity is 3 invocations of the multiplication protocol. On the other hand, the parties can also compute $[ab]_p$ and $[cd]_p$ in parallel, and finally compute $[abcd]_p$. Then the complexity of this computation is 2 rounds and 3 invocations. We will evaluate the round complexity of a protocol by performing the multiplication protocol in parallel as much as possible.

3.3.4 Bitwise Sharing

The concept of bitwise sharing is to share $a \in \mathbb{Z}_p (= \{0, 1, \ldots, p - 1\})$ in the form of $\{[a_{\ell-1}]_p, \ldots, [a_0]_p\}$ such that $a = \sum_{i=0}^{\ell-1} 2^i a_i$ where $a_i \in \{0, 1\}$. We use $[a]_B$ to denote $\{[a_{\ell-1}]_p, \ldots, [a_1]_p, [a_0]_p\}$.

3.3.5 Subprotocols

We describe several subprotocols in [BB89, DFK+06] necessary for our constructions. All these subprotocols run in a constant number of rounds. By combining these subprotocols, we will construct our interval test, equality test, comparison, and bit-decomposition protocols that also run in a constant number of rounds.

3.3.5.1 Joint Random Number Sharing

The parties can share a uniformly random, unknown number r [BB89] as follows: Each P_i picks up $r_{\_i} \in \mathbb{Z}_p$ at random and shares it by a sharing $[r_{\_i}]_p = \{f_i(1), \ldots, f_i(n)\}$ where $f_i(0) = r_{\_i}$ and f_i is a random polynomial. That is, P_i distributes $f_i(j)$'s to other P_j's. From each $[r_{\_i}]_p$, the parties compute $[r]_p = \sum_{i=1}^{n} [r_{\_i}]_p$. We assume that the complexity for this is almost the same as the complexity of 1 invocation of the multiplication protocol. We denote this subprotocol as $[r \in_R \mathbb{Z}_p]_p$.

3.3.5.2 Joint Random Bit Sharing

The parties can share a uniformly random $a \in \{0, 1\}$ as follows: The parties compute $[r \in_R \mathbb{Z}_p]_p$, perform the multiplication protocol to obtain $[r^2]_p$ and reveal r^2. If $r^2 = 0$, the parties retry. If $r^2 \neq 0$, the parties compute $r' = \sqrt{r^2}$ such that $0 < r' < \frac{p}{2}$. This can be done in polynomial time because p is an odd prime. Then the parties set $[a]_p = 2^{-1}(r'^{-1}[r]_p + 1)$. It is clear that $r'^{-1}r \in \{-1, 1\}$; hence $a \in \{0, 1\}$. The total complexity is 2 rounds and 2 invocations. We denote this subprotocol as $[a \in_R \{0, 1\}]_p$. In the threshold homomorphic setting [CDN01, DN03], this can be computed as $a = \oplus_{i=1}^{n} b_i$ where $b_i \in_R \{0, 1\}$ is generated by P_i (see [ST06] for the details).

3.3.5.3 Unbounded Fan-In Or

Given $[a_{\ell-1}]_p, \ldots, [a_0]_p$ where $a_i \in \{0, 1\}$, the parties can compute $[\vee_{i=0}^{\ell-1} a_i]_p$ in a constant number of rounds. For this, as in [DFK+06], we can use the same technique to evaluate symmetric Boolean functions as follows:

The parties compute $[A]_p = 1 + \sum_{i=0}^{\ell-1} [a_i]_p$. Note that $1 \leq A \leq \ell + 1$. Next, the parties define a ℓ-degree polynomial $f_\ell(x)$ such that $f_\ell(1) = 0$ and $f_\ell(2) = f_\ell(3) = \cdots = f_\ell(\ell + 1) = 1$. $f_\ell(x)$

can be determined by using Lagrange interpolation. Note that $f_\ell(A) = \vee_{i=0}^{\ell-1} a_i$. Then the parties try to obtain $[\vee_{i=0}^{\ell-1} a_i]_p$ by computing $[f_\ell(A)]_p$ from $[A]_p$ and $f_\ell(x)$. This can be done in a constant number of rounds by using an unbounded fan-in multiplication and the inversion protocol [BB89] as follows:

Let's assume that $f_\ell(x)$ is represented as $f_\ell(x) = \alpha_0 + \alpha_1 x + \cdots + \alpha_\ell x^\ell \bmod p$. To obtain $[f_\ell(A)]_p$, the parties compute $[A]_p, [A^2]_p, \ldots, [A^\ell]_p$ because $[f_\ell(A)]_p = \alpha_0 + \sum_{i=1}^{\ell} \alpha_i [A^i]_p$.

For $1 \le i \le \ell$, the parties generate $[b_i \in_R \mathbb{Z}_p]_p$ and $[b'_i \in_R \mathbb{Z}_p]_p$ in parallel, compute $[B_i]_p = [b_i]_p \times [b'_i]_p$, and reveal B_i. Note that $[b_i^{-1}]_p$ can be computed as $[b_i^{-1}]_p = B_i^{-1}[b'_i]_p$ at the same time (inversion protocol).

Next, the parties compute in parallel

$$
\begin{aligned}
[c_1]_p &= [A]_p \times [b_1^{-1}]_p \\
[c_2]_p &= [A]_p \times [b_1]_p \times [b_2^{-1}]_p \\
[c_3]_p &= [A]_p \times [b_2]_p \times [b_3^{-1}]_p \\
&\vdots \\
[c_{\ell-1}]_p &= [A]_p \times [b_{\ell-2}]_p \times [b_{\ell-1}^{-1}]_p \\
[c_\ell]_p &= [A]_p \times [b_{\ell-1}]_p \times [b_\ell^{-1}]_p
\end{aligned}
$$

and reveal all c_i's.

Then the parties can compute $[A^i]_p = (\prod_{k=1}^{i} c_k)[b_i]_p$.

If $A = 0$, information about A is leaked. That is why we used $[A]_p = 1 + \sum_{i=0}^{\ell-1} [a_i]_p$ to guarantee that A is not zero.

The complexity of computing each component is as follows: 2 rounds and 3ℓ invocations for $[b_i]_p$'s, $[b'_i]_p$'s, and B_i's and 2 rounds and 2ℓ invocations for c_i's. $[b_i]_p \times [b_{i+1}^{-1}]_p$ for $1 \le i \le \ell - 1$ can be precomputed as $[b_i]_p \times [b'_{i+1}]_p$ in the second round in parallel with $[b_i]_p \times [b'_i]_p$. Therefore, the total complexity is 3 rounds (including 2 rounds for random value generation) and 5ℓ invocations.

Note that we can compute unbounded fan-in And and Xor similarly because a symmetric Boolean function depends only on the number of 1's in its inputs. Also note that the random values necessary for this protocol can be generated in advance rather than on demand when this

subprotocol is used as a building block in the larger protocol, thus reducing the round complexity. Actually all the random value generation (for bits and numbers) can be done in the first 2 rounds (3 rounds in the setting [ST06] by using an unbounded fan-in Xor).

3.3.5.4 Prefix-Or

Given $[a_1]_p, \ldots, [a_\ell]_p$ where $a_i \in \{0, 1\}$, the parties can compute the Prefix-Or $[b_1]_p, \ldots, [b_\ell]_p$ such that $b_i = \vee_{j=1}^{i} a_j$ in a constand number of rounds. As in [DFK+06], this can be done by using the technique from [CFL83a] as follows:

For notational convenience, let's assume that $\ell = \lambda^2$ for an integer λ and index the bits a_k as $a_{i,j} = a_{\lambda(i-1)+j}$ for $i, j = 1, \ldots, \lambda$. Other cases can be adapted quite straightforwardly.

First the parties compute $[x_i]_p = \vee_{j=1}^{\lambda} [a_{i,j}]_p$ for $i = 1, \ldots, \lambda$ in parallel by using unbounded fan-in Or where the size of problems is λ instead of ℓ. Then the parties compute similarly $[y_i]_p = \vee_{k=1}^{i} [x_k]_p$ for $i = 1, \ldots, \lambda$ in parallel. Now we can notice that $y_i = 1$ iff some block $\{a_{i',1}, \ldots, a_{i',\lambda}\}$ with $i' \leq i$ contains a $a_{i',j} = 1$.

Next, the parties set $[f_1]_p = [x_1]_p$, and for $i = 2, \ldots, \lambda$, set $[f_i]_p = [y_i]_p - [y_{i-1}]_p$. Now we can notice that $f_i = 1$ iff $\{a_{i,1}, \ldots, a_{i,\lambda}\}$ is the first block containing a $a_{i,j} = 1$. Let i_0 be such that $f_{i_0} = 1$. The parties can compute $\{[a_{i_0,1}]_p, \ldots, [a_{i_0,\lambda}]_p\}$ by $[a_{i_0,j}]_p = \sum_{i=1}^{\lambda} [f_i]_p \times [a_{i,j}]_p$ in parallel without revealing i_0.

Next, the parties compute $\{[b_{i_0,1}]_p, \ldots, [b_{i_0,\lambda}]_p\}$ where $b_{i_0,j} = \vee_{k=1}^{j} a_{i_0,k}$ by using unbounded fan-in Or in parallel.

Finally, the parties set $[s_i]_p = [y_i]_p - [f_i]_p$. Then $s_i = 1$ iff $i > i_0$. If we index the bits of Prefix-Or b_k as $b_{i,j} = b_{\lambda(i-1)+j}$ as we did for a_k, the Prefix-Or can be computed as $[b_k]_p = [b_{\lambda(i-1)+j}]_p = [b_{i,j}]_p = [f_i]_p \times [b_{i_0,j}]_p + [s_i]_p$ in the end.

When we use several invocations of unbounded fan-in Or, all the necessary random values in unbounded fan-in Or can be generated in the first 2 rounds. Therefore, the total complexity is 7 rounds (including 2 rounds for random value generation) and 17ℓ invocations. [*1] Similarly the Prefix-And can also be computed by using the same technique.

[*1] The evaluation in [DFK+06] is 17 rounds and 20ℓ invocations by generating random values on demand.

3.3.5.5 Bitwise Less-Than

Given two bitwise sharings $[a]_B$ and $[b]_B$, the parties can compute $[a < b]_p$ without revealing $(a < b)$ itself. The basic idea is the same as the circuit for the millionaire's problem. We will give an outline of this subprotocol based on the description in [DFK+06].

For $0 \leq i \leq \ell-1$, the parties compute $[c_i]_p = [a_i \oplus b_i]_p = [a_i]+[b_i]_p-2[a_i b_i]_p$ in parallel and then compute $[d_i]_p = \vee_{j=i}^{\ell-1}[c_j]_p$ by using Prefix-Or, and set $[e_i]_p = [d_i - d_{i+1}]_p$ where $[e_{\ell-1}]_p = [d_{\ell-1}]_p$. Finally, the parties compute $[a < b]_p = \sum_{i=0}^{\ell-1}([e_i]_p \times [b_i]_p)$ in parallel.

The complexity of computing each component is as follows: 1 round and ℓ invocations for c_i's, 7 rounds and 17ℓ invocations for the Prefix-Or, and 1 round and ℓ invocations for $\sum_{i=0}^{\ell-1}([e_i]_p \times [b_i]_p)$. Because c_i's can be computed in parallel with random value generation in the Prefix-Or, the total complexity is 8 rounds (including 2 rounds for random value generation) and 19ℓ invocations. We use $[a <_B b]_p$ in order to stress that a and b are bitwise-shared. Note that if b is known, the complexity is 7 rounds (including 2 rounds for random value generation) and 17ℓ invocations by saving the invocations for c_i's and $\sum_{i=0}^{\ell-1}([e_i]_p \times [b_i]_p)$.

3.3.5.6 Joint Random Number Bitwise-Sharing

The parties can bitwise-share a uniformly random, unknown number r such that $0 \leq r = \sum_{i=0}^{\ell-1} 2^i r_i < p$ as follows: The parties generate each bit, $[r_i \in_R \{0,1\}]_p$ for $0 \leq i \leq \ell-1$ in parallel, compute $[r <_B p]_p$ by using the bitwise less-than protocol and reveal $(r < p)$. If $r \geq p$, the parties retry.

The complexity of computing each component is as follows: 2 rounds and 2ℓ invocations for r_i's and 7 rounds and 17ℓ invocations for the bitwise less-than protocol (note that p is known). Because r_i's can be generated in parallel with random value generation in the Prefix-Or of the bitwise less-than protocol, the complexity is 7 rounds and 19ℓ invocations. As in [DFK+06], we assume that at least one of four generated candidates is less than p and the amortized complexity is 7 rounds (including 2 rounds for random value generation) and 76ℓ invocations. We denote this subprotocol as $[r \in_R \mathbb{Z}_p]_B$.

3.3.5.7 Bitwise Sum

Given two bitwise sharings $[a]_B = \{[a_{\ell-1}]_p, \ldots, [a_0]_p\}$ and $[b]_B = \{[b_{\ell-1}]_p, \ldots, [b_0]_p\}$, the parties can compute the bitwise sharing $[d]_B = \{[d_\ell]_p, \ldots, [d_0]_p\}$ such that $d = a + b$ over the integers (not mod p). By using the method of [CFL83b], the bitwise sum protocol can be performed in constant rounds (see [DFK+06] for the details).

Based on [DFK+06] (see unbounded fan-in carry propagation in Sect. 6.4), the complexity of the bitwise sum protocol is evaluated as follows: If the Prefix-And is computed with x rounds and $y \times \ell$ invocations, the complexity of the bitwise sum protocol is upper bounded by $2(x + 1) + 1$ rounds and $(2(y + 6) + 1)\ell \log_2 \ell$ invocations. Assuming that all the random values are generated in the first 2 rounds and that the complexity of the Prefix-And is 5 rounds (not including 2 rounds for random value generation) and 17ℓ invocations, the total complexity is 15 rounds (including 2 rounds for random value generation) and $47\ell \log_2 \ell$ invocations. [*2]

We denote this subprotocol as $[d]_B = [a]_B + [b]_B$.

3.4 Existing Protocols [DFK+06, ST06]

Damgård *et al.* [DFK+06] have shown a novel technique to convert $[a]_p$ into $[a]_B$. This technique is called the bit-decomposition protocol (Fig. 3.1). Note that we can obtain $[a]_p$ from $[a]_B$ easily by computing $[a]_p = \sum_{i=0}^{\ell-1} 2^i [a_i]_p \bmod p$. Also, Schoenmakers and Tuyls [ST06] have proposed a similar bit-decomposition protocol (called **BITREP** gate) in the context of multiparty computation [CDN01, DN03] based on threshold additively-homomorphic cryptosystems.

The complexity of computing each component in [DFK+06] is as follows: 7 rounds (including 2 rounds for random value generation) and 76ℓ invocations for $[r \in_R \mathbb{Z}_p]_B$, 13 rounds and $47\ell \log_2 \ell$ invocations for $[d]_B$ (bitwise sum), 5 rounds and 17ℓ invocations for $[q]_p$, that is, $[d <_B p]_p$, and 13 rounds and $47\ell \log_2 \ell$ invocations for $[h]_B$ (bitwise sum). The total complexity is 38 rounds (including 2 rounds for random value generation) and $93\ell + 94\ell \log_2 \ell$ invocations.

By using the bit-decomposition protocol, any bit-oriented operation can be performed in arith-

[*2] The evaluation in [DFK+06] is 37 rounds and $55\ell \log_2 \ell$ invocations by generating random values on demand.

The parties convert $[a]_p$ into $[a]_B$.

1. The parties generate $[r]_B$ and obtain $[r]_p$ eventually.
2. The parties compute $[c]_p = [a]_p - [r]_p$ and reveal $c = a - r \bmod p \in \{0, 1, \ldots, p-1\}$.
3. The parties compute $[d]_B = [r]_B + [c]_B = \{[d_\ell]_p, \ldots, [d_0]_p\}$.
4. Note that d can be represented as $d = a + qp$ where $q \in \{0, 1\}$. The parties can compute the bit q as $[q]_p = [p \leq d]_p = 1 - [d <_B p]_p$.
5. Consider $g = (2^\ell - qp) \bmod 2^\ell$ and its bitwise sharing $[g]_B = \{[g_{\ell-1}]_p, \ldots, [g_0]_p\}$. Let $(f_{\ell-1}, \ldots, f_0)_2$ be the bit representation of $2^\ell - p$ such that $2^\ell - p = \sum_{i=0}^{\ell-1} 2^i f_i$ and $f_i \in \{0, 1\}$. Then the parties can compute $[g]_B$ by $[g_i]_p = f_i[q]_p$ for $0 \leq i \leq \ell - 1$ because $g = 0$ if $q = 0$ and $g = 2^\ell - p$ if $q = 1$.
6. The parties now have the two following bitwise sharings, $[d]_B = [a+qp]_B$ and $[g]_B = [(2^\ell - qp) \bmod 2^\ell]_B$. Therefore, the parties can compute $[h]_B = [d]_B + [g]_B$ where $h = a + q2^\ell$.
7. By discarding the sharing $[h_\ell]_p$ from $[h]_B$, they can obtain $[a]_B$.

Figure. 3.1 Bit-Decomposition [DFK+06]

metic circuits where inputs are given as polynomial sharings (rather than bitwise sharings) of elements in $\mathbb{Z}_p$.

However, the bit-decomposition protocol is not cheap, so we try to construct a simplified bit-decomposition protocol and construct more efficient protocols for interval tests, equality tests, and comparisons without relying on the bit-decomposition protocol.

3.5 Simplified Bit-Decomposition Protocol

In the original bit-decomposition protocol, we need 2 invocations of the bitwise sum protocol (in Steps 3 and 6 in Fig. 3.1). We can notice that the first invocation for $[d]_B$ can be eliminated by changing the way in which we compute $[q]_p$ based on the following observation.

In Step 4 of the original protocol, the parties compute $[q]_p = 1 - [d <_B p]_p$ where $d = r + c$, c is public, and r is bitwise-shared. Therefore, the condition, $(d < p)$ can be changed into $(r < p - c)$.

The parties convert $[a]_p$ into $[a]_B$.

1. The parties generate $[r]_B$ and obtain $[r]_p$ eventually.
2. The parties compute $[c]_p = [a]_p - [r]_p$ and reveal $c = a - r \bmod p \in \{0, 1, \ldots, p-1\}$. If $c = 0$, the parties are successfully done because $[r]_B$ is equal to $[a]_B$ by a coincidence.
3. If $c \neq 0$, next, the parties compute the bit q, $[q]_p = [p \leq r + c]_p = 1 - [r <_B p - c]_p$ by using the bitwise less-than protocol.
4. Note that a can be represented as $a = c + r - qp$ over the integers where $q \in \{0, 1\}$. Therefore, we also have $2^\ell + a = 2^\ell + c - qp + r$ over the integers. Consider $\widetilde{g} = (2^\ell + c - qp) \bmod 2^\ell$ and its bitwise sharing $[\widetilde{g}]_B = \{[\widetilde{g}_{\ell-1}]_p, \ldots, [\widetilde{g}_0]_p\}$. Let $(\widetilde{f}_{\ell-1}, \ldots, \widetilde{f}_0)_2$ be the bit representation of $2^\ell + c - p$ such that $2^\ell + c - p = \sum_{i=0}^{\ell-1} 2^i \widetilde{f}_i$ and $\widetilde{f}_i \in \{0, 1\}$. Also, let $(\widetilde{f}'_{\ell-1}, \ldots, \widetilde{f}'_0)_2$ be the bit representation of c such that $c = \sum_{i=0}^{\ell-1} 2^i \widetilde{f}'_i$ and $\widetilde{f}'_i \in \{0, 1\}$. Then the parties can compute $[\widetilde{g}]_B$ by $[\widetilde{g}_i]_p = (\widetilde{f}_i - \widetilde{f}'_i)[q]_p + \widetilde{f}'_i$ for $0 \leq i \leq \ell - 1$ because $\widetilde{g} = c$ if $q = 0$ and $\widetilde{g} = 2^\ell + c - p$ if $q = 1$.
5. The parties now have the two following bitwise sharings, $[r]_B$ and $[\widetilde{g}]_B = [(2^\ell + c - qp) \bmod 2^\ell]_B$. Therefore, the parties can compute $[h]_B = [r]_B + [\widetilde{g}]_B$ where $h = a + q2^\ell$.
6. By discarding the sharing $[h_\ell]_p$ from $[h]_B$, they can obtain $[a]_B$.

Figure. 3.2 Simplified Bit-Decomposition

The parties have $[r]_B$ and $p - c$ is public, so $(r < p - c)$ can be computed by using the bitwise less-than protocol without computing $[d]_B = [r]_B + [c]_B$, thus eliminating one invocation of the bitwise sum protocol.

Since we have eliminated $[d]_B$, we need to specify how to compute $[a]_B$ in the rest of the protocol. Fortunately, we can use $[r]_B$ itself to compute $[a]_B$ by using the bitwise sum protocol. The simplified bit-decomposition protocol is given in Fig. 3.2.

3.5.1 Complexity of Bit-Decomposition Protocol

The complexity of computing each component is as follows: 7 rounds (including 2 rounds for random value generation) and 76ℓ invocations for $[r \in_R \mathbb{Z}_p]_B$, 5 rounds and 17ℓ invocations for

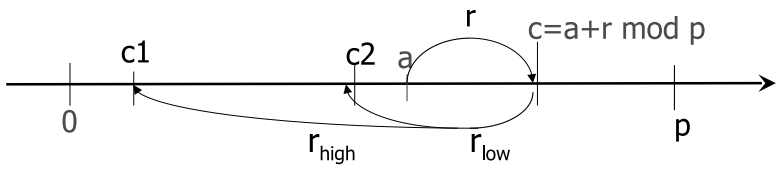

Figure. 3.3 Case of $c_2 \leq c$

$[q]_p$, that is, $[r <_B p - c]_p$, and 13 rounds and $47\ell \log_2 \ell$ invocations for $[h]_B$. The total complexity is 25 rounds (including 2 rounds for random value generation) and $93\ell + 47\ell \log_2 \ell$ invocations.

3.6 Proposed Protocols Without Bit-Decomposition

3.6.1 Interval Test Protocol

In the interval test protocol, given public constants $c_1, c_2 \in \mathbb{Z}_p$ (where $c_1 < c_2$) and shared secret $a \in \mathbb{Z}_p$, the parties compute $[c_1 < a < c_2]_p$ without revealing $(c_1 < a < c_2)$ itself.

If the parties use the bit-decomposition protocol, the parties compute $[a]_B$ from $[a]_p$ and compute $[c_1 < a < c_2]_p = [c_1 <_B a]_p \times [a <_B c_2]_p$.

The basic idea of our construction is as follows: We randomize a by $c = a + r$ and reveal c where r is a bitwise-shared random secret. We obtain an appropriate interval $[r_{\text{low}}, r_{\text{high}}]$ from c, c_1, and c_2. Then computing $[c_1 < a < c_2]_p$ is reduced to checking whether r exists in the appropriate interval $r_{\text{low}} < r < r_{\text{high}}$ (for example, see Fig. 3.3) by the bitwise less-than protocol.

3.6.1.1 Procedure

The parties generate $[r \in_R \mathbb{Z}_p]_B$ and obtain $[r]_p$ eventually. Next, the parties compute $[c]_p = [a]_p + [r]_p$ and reveal $c = a + r \bmod p \in \{0, 1, \ldots, p-1\}$. At this point, no information about a is leaked from c because r is uniformly random and unknown to the parties. Now we can think that $a \in \{-(p-c-1), \ldots, -1, 0, 1, \ldots, c-1, c\}$ because $r \in \{0, 1, \ldots, p-1\}$.

First, we consider the case where $c_1 < c < c_2$ does not hold. When $c_2 \leq c$ (see Fig. 3.3), obviously, we have $(c_1 < a < c_2) = 1$ if $(r_{\text{low}} =) c - c_2 < r < c - c_1 (= r_{\text{high}})$. Similarly, when $c \leq c_1$ (see Fig. 3.4), if $(r_{\text{low}} =) c + p - c_2 < r < c + p - c_1 (= r_{\text{high}})$, we have $-(p - c_1) < a < -(p - c_2)$.

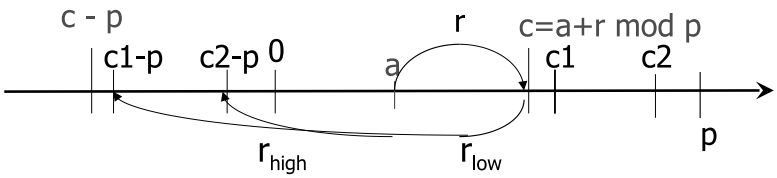

Figure. 3.4 Case of $c \le c_1$

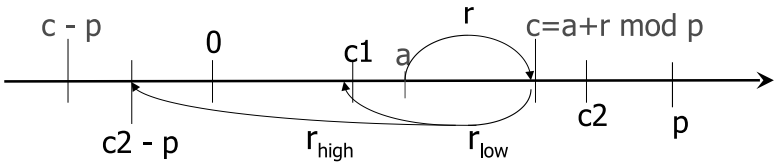

Figure. 3.5 Case of $c_1 < c < c_2$

This means that $(c_1 < a < c_2) = 1$. Therefore, the parties compute, by using the bitwise less-than protocol,

$$[c_1 < a < c_2]_p = [r_{\text{low}} <_B r]_p \times [r <_B r_{\text{high}}]_p.$$

Next, we consider the case where $c_1 < c < c_2$ holds (see Fig. 3.5). In this case, if $(r_{\text{low}} =)c - c_1 - 1 < r < c + p - c_2 + 1 (= r_{\text{high}})$, we have $-(p - c_2) \le a \le c_1$. This means that $(c_1 < a < c_2) = 0$. Therefore, the parties compute

$$[r_{\text{low}} < r < r_{\text{high}}]_p = [c - c_1 - 1 <_B r]_p \times [r <_B c + p - c_2 + 1]_p$$

by using the bitwise less-than protocol and set

$$[c_1 < a < c_2]_p = 1 - [r_{\text{low}} < r < r_{\text{high}}]_p.$$

3.6.1.2 Complexity of Interval Test Protocol

If we use the bit-decomposition protocol straightforwardly, the complexity of computing each component is as follows: 38 rounds (including 2 rounds for random value generation) and $93\ell + 94\ell \log_2 \ell$ invocations for $[a]_B$, 5 rounds and $(17\ell \times 2)$ invocations for $[c_1 <_B a]_p$ and $[a <_B c_2]_p$, and 1 round and 1 invocation for $[c_1 <_B a]_p \times [a <_B c_2]_p$. The total complexity is 44 rounds (including 2 rounds for random value generation) and $127\ell + 94\ell \log_2 \ell + 1$ invocations.

On the other hand, in our construction, the complexity of computing each component is as follows: 7 rounds (including 2 rounds for random value generation) and 76ℓ invocations for $[r \in_R \mathbb{Z}_p]_B$, 5 rounds and $(17\ell \times 2)$ invocations for $[r_{\text{low}} <_B r]_p$ and $[r <_B r_{\text{high}}]_p$, and 1 round and 1 invocation for $[r_{\text{low}} <_B r]_p \times [r <_B r_{\text{high}}]_p$. The total complexity is 13 rounds (including 2 rounds for random value generation) and $110\ell + 1$ invocations.

3.6.2 LSB Protocol for Special Case of Interval Test Protocol

In order to construct our comparison protocol later, we consider computing $[a < \frac{p}{2}]_p$. Though it is possible for us to use the technique in Sect. 3.6.1, we compute $[a < \frac{p}{2}]_p$ more efficiently by using special properties of $\frac{p}{2}$ and apply this subprotocol (called the LSB protocol here) to our comparison protocol. By a simple observation, we can notice that

$$a \in \{0, 1, \ldots, \frac{p-1}{2}\} \Leftrightarrow (2a \bmod p)_0 = 0,$$

and that

$$a \in \{\frac{p-1}{2} + 1, \ldots, p-1\} \Leftrightarrow (2a \bmod p)_0 = 1$$

where $(x)_0$ is the least significant bit (LSB) of $x \in \{0, 1, \ldots, p-1\}$. That is, if $a < \frac{p}{2}$, *no* wrap-around modulo p occurs when $2a \bmod p$ is computed and $2a \bmod p$ is even. On the other hand, if $a > \frac{p}{2}$, a wrap-around modulo p occurs when $2a \bmod p$ is computed and $2a \bmod p$ is odd. Therefore, if we can compute $[(x)_0]_p$ from $[x]_p$, we can use it to compute $[a < \frac{p}{2}]_p$.

To compute $[(x)_0]_p$ from $[x]_p$, we randomize x by $c = x + r$ and reveal c where r is a bitwise-shared random secret. Then we can obtain $[(x)_0]_p$ from $(c)_0$ and $[(r)_0]_p$.

3.6.2.1 Procedure

The parties want to compute $[(x)_0]_p$ from $[x]_p$. The parties generate $[r \in_R \mathbb{Z}_p]_B$ and obtain $[r]_p$ eventually. Next, the parties compute $[c]_p = [x]_p + [r]_p$ and reveal $c = x + r \bmod p \in \{0, 1, \ldots, p-1\}$. If *no* wrap-around modulo p occurs when c is computed, we have $(x)_0 = (c)_0 \oplus (r)_0$ and if a wrap-around modulo p occurs when c is computed, we have $(x)_0 = 1 - \{(c)_0 \oplus (r)_0\}$. Furthermore, we can use $(c < r)$ to know whether or not a wrap-around modulo p occurred when c was computed. That is, if $(c < r) = 0$, it means that no wrap-around modulo p occurred, and if

$(c < r) = 1$, it means that a wrap-around modulo p occurred because $r \in \{0, 1, \ldots, p-1\}$.

From these facts, the parties can compute $[(x)_0]_p$ as

$$\begin{aligned}[(x)_0]_p &= [c <_B r]_p \times (1 - \{(c)_0 \oplus [(r)_0]_p\}) + (1 - [c <_B r]_p) \times \{(c)_0 \oplus [(r)_0]_p\} \\ &= [c <_B r]_p + \{(c)_0 \oplus [(r)_0]_p\} - 2[c <_B r]_p \times \{(c)_0 \oplus [(r)_0]_p\}. \end{aligned} \tag{3.1}$$

The interpretation of Eq. (3.1) is that if $(c <_B r) = 1$, we have $(1 - \{(c)_0 \oplus [(r)_0]_p\})$ and otherwise we have $\{(c)_0 \oplus [(r)_0]_p\}$. Because c is public, note that $(c)_0 \oplus [(r)_0]_p$ can be computed as

$$(c)_0 \oplus [(r)_0]_p = \begin{cases} [(r)_0]_p & \text{if } (c)_0 = 0 \\ 1 - [(r)_0]_p & \text{if } (c)_0 = 1. \end{cases}$$

Also note that the parties already have $[(r)_0]_p$ because r is generated by $[r \in_R \mathbb{Z}_p]_B$.

By using the LSB protocol, the parties can compute $[a < \frac{p}{2}]_p$ from $[a]_p$ as

$$[a < \frac{p}{2}]_p = 1 - [(2a)_0]_p.$$

3.6.2.2 Complexity of LSB Protocol

The complexity of computing each component is as follows: 7 rounds (including 2 rounds for random value generation) and 76ℓ invocations for $[r \in_R \mathbb{Z}_p]_B$, 5 rounds and 17ℓ invocations for $[c <_B r]_p$, and 1 round and 1 invocation for $[c <_B r]_p \times [(r)_0]_p$. The total complexity is 13 rounds (including 2 rounds for random value generation) and $93\ell + 1$ invocations.

3.6.3 Comparison Protocol

In the comparison protocol, given two shared secrets $a, b \in \mathbb{Z}_p$, the parties compute $[a < b]_p$ without revealing $(a < b)$ itself. For example, we can compute $[\max(a, b)]_p = [a]_p + [a < b]_p \times [b - a]_p$ by using the comparison protocol.

If the parties use the bit-decomposition protocol, the parties compute $[a]_B$ and $[b]_B$ from $[a]_p$ and $[b]_p$ and compute $[a <_B b]_p$ as in [DFK+06].

It seems difficult for us to compare a and b directly without using the bit-decomposition protocol. Therefore, we compare a and b indirectly via the value of $\frac{p}{2}$ by computing $[a < \frac{p}{2}]_p$, $[b < \frac{p}{2}]_p$, and $[a - b \bmod p < \frac{p}{2}]_p$.

Table. 3.2 Truth Table for $(a < b)$

$w = (a < p/2)$	$x = (b < p/2)$	$y = (a - b \bmod p < p/2)$	$z = (a < b)$
1	0	*	1
0	1	*	0
0	0	0	1
0	0	1	0
1	1	0	1
1	1	1	0

3.6.3.1 Procedure

By a simple observation, we can notice that $(a < b)$ is determined from $(a < \frac{p}{2})$, $(b < \frac{p}{2})$, and $(a - b \bmod p < \frac{p}{2})$. This observation can be confirmed by the truth table (Table 3.2).

When we denote $(a < \frac{p}{2})$, $(b < \frac{p}{2})$, $(a - b \bmod p < \frac{p}{2})$, and $(a < b)$ as w, x, y, and z respectively, then z is represented as

$$\begin{aligned} z &= w\overline{x} \vee \overline{w}\overline{x}\overline{y} \vee wx\overline{y} \\ &= w(1-x) + (1-w)(1-x)(1-y) + wx(1-y) \\ &= w(x + y - 2xy) + 1 - y - x + xy. \end{aligned} \tag{3.2}$$

Therefore, if the parties can compute $[a < \frac{p}{2}]_p, [b < \frac{p}{2}]_p$, and $[a - b \bmod p < \frac{p}{2}]_p$, they can compute $[a < b]_p$ from Eq. (3.2) by using addition and the multiplication protocol. We can use the LSB protocol to compute all three of these values.

3.6.3.2 Complexity of Comparison Protocol

If we use the bit-decomposition protocol straightforwardly, the complexity of computing each component is as follows: 38 rounds (including 2 rounds for random value generation) and $2 \times (93\ell + 94\ell \log_2 \ell)$ invocations for $[a]_B$ and $[b]_B$ and 6 rounds and 19ℓ invocations for $[a <_B b]_p$. The total complexity is 44 rounds (including 2 rounds for random value generation) and $205\ell + 188\ell \log_2 \ell$ invocations.

On the other hand, in our construction, the complexity of computing each component is as follows: 13 rounds (including 2 rounds for random value generation) and $3\times(93\ell+1)$ invocations for $[a < \frac{p}{2}]_p$, $[b < \frac{p}{2}]_p$, and $[a - b \bmod p < \frac{p}{2}]_p$ and 2 rounds and 2 invocations for Eq. (3.2). The total complexity is 15 rounds (including 2 rounds for random value generation) and $279\ell + 5$ invocations.

3.6.4 Equality Test Protocol

In the equality test protocol, given two shared secrets $a, b \in \mathbb{Z}_p$, the parties compute $[a = b]_p$ without revealing $(a = b)$ itself.

Because $[a = b]_p$ can be computed by $[a - b = 0]_p$, we focus on computing $[a = 0]_p$.

If the parties use the bit-decomposition protocol, the parties compute $[d]_B$ from $[d]_p = [a - b]_p$ and compute $[\wedge_{i=0}^{\ell-1}(1 - d_i)]_p$ by using an unbounded fan-in And as in [DFK+06]. In the secret sharing setting, it is also possible to use secure exponentiation $[d = 0]_p = 1 - [d]_p^{p-1}$ as in [CD01] but this requires $O(\log_2 p)$ rounds. In [BB89], the equality test protocol called normalization was proposed to compute extended inverses, but it requires $O(p^2)$ invocations and it will be impractical for large p.

In our construction, we use a very simple observation that the randomization $c(= d + r)$ of d is equal to r if d is zero.

3.6.4.1 Procedure

First the parties generate $[r \in_R \mathbb{Z}_p]_B$ and obtain $[r]_p$ eventually. Next, the parties compute $[c]_p = [a]_p + [r]_p$ and reveal $c = a + r \bmod p \in \{0, 1, \ldots, p - 1\}$. We can note that $c = r$ iff $a = 0$. Therefore, the parties compute whether all bits of c are the same as $[r]_B$. Let $(c_{\ell-1}, \ldots, c_0)_2$ be the bit representation of c. Then the parties compute $[c'_i]_p$ for $0 \le i \le \ell - 1$ as

$$[c'_i]_p = \begin{cases} [r_i]_p & \text{if } c_i = 1 \\ 1 - [r_i]_p & \text{if } c_i = 0. \end{cases}$$

We can note that $c'_i \in \{0, 1\}$ and that $c'_i = 1$ iff $c_i = r_i$. Finally, the parties compute $[a = 0]_p$ as $[\wedge_{i=0}^{\ell-1} c'_i]_p$ by using an unbounded fan-in And.

3.6.4.2 Complexity of Equality Test Protocol

If we use the bit-decomposition protocol straightforwardly, the complexity of computing each component is as follows: 38 rounds (including 2 rounds for random value generation) and $93\ell + 94\ell \log_2 \ell$ invocations for $[d]_B$ and 1 rounds and 5ℓ invocations for $[\wedge_{i=0}^{\ell-1}(1 - d_i)]_p$. The total complexity is 39 rounds (including 2 rounds for random value generation) and $98\ell + 94\ell \log_2 \ell$ invocations.

On the other hand, in our construction, the complexity of computing each component is as follows: 7 rounds (including 2 rounds for random value generation) and 76ℓ invocations for $[r]_B$ and 1 rounds and 5ℓ invocations for $[\wedge_{i=0}^{\ell-1} c'_i]_p$. The total complexity is 8 rounds (including 2 rounds for random value generation) and 81ℓ invocations.

3.6.5 Probabilistic Equality Test Protocol

We consider another version of the equality test protocol with a very small round complexity. We focus on computing $[a = 0]_p$ again. In our construction, we assume that $p = 3 \bmod 4$ or $p = 5 \bmod 8$. These imply that Legendre symbol $\left(\frac{-1}{p}\right) = -1$ if $p = 3 \bmod 4$ and that $\left(\frac{2}{p}\right) = -1$ if $p = 5 \bmod 8$. The basic idea is based on the property of quadratic residues as follows: If a is a zero, we always have $\left(\frac{c}{p}\right) = \left(\frac{r}{p}\right)$ where $c = a + r$, r is a random secret and c is a revealed value. If a is not a zero, we have $\left(\frac{c}{p}\right) \neq \left(\frac{r}{p}\right)$ with non-negligible probability. By checking whether $\left(\frac{c}{p}\right) = \left(\frac{r}{p}\right)$ secretly with sufficiently many trials, we can perform the equality test on a in a probabilistic way. Here note that we need to generate random secret r in a special way to compute $\left(\frac{r}{p}\right)$ secretly.

3.6.5.1 Procedure

First we describe the case of $p = 3 \bmod 4$. The case of $p = 5 \bmod 8$ can be obtained quite straightforwardly as we mention later.

The parties generate $[b_j \in_R \{-1, 1\}]_p$, $[r_j \in_R \mathbb{Z}_p]_p$, and $[r'_j \in_R \mathbb{Z}_p]_p$ for $1 \leq j \leq k$ in parallel where k is chosen such that the error probability $\left(\frac{1}{2}\right)^k$ is negligible. The value b_j can be generated by a joint random bit sharing. Next, the parties compute for $1 \leq j \leq k$ in parallel,

$$[c_j]_p = [a]_p \times [r_j]_p + [b_j]_p \times [r'_j]_p \times [r'_j]_p$$

and reveal all the c_j's. Note that $b_j r_j'^2$ is uniformly random and unknown to the parties, so no information about a is leaked from c_j.

Actually we can confirm the probabilities as follows:

$$\Pr[b_j r_j'^2 = 0] = \Pr[r_j' = 0] = \frac{1}{p};$$

$$\Pr[b_j r_j'^2 = y] = \Pr[b_j = 1] \times \Pr[r_j' = \pm\sqrt{y}] = \frac{1}{2} \times \frac{2}{p} = \frac{1}{p} \text{ if } y \text{ is a quadratic residue;}$$

$$\Pr[b_j r_j'^2 = y] = \Pr[b_j = -1] \times \Pr[r_j' = \pm\sqrt{-y}] = \frac{1}{2} \times \frac{2}{p} = \frac{1}{p} \text{ if } y \text{ is a quadratic nonresidue.}$$

Also note that if $a = 0$, ar_j is always a zero and that if $a \neq 0$, ar_j is uniformly random.

If c_j is a zero, the parties discard the c_j and retry. The probability that c_j happens to be a zero is $\frac{1}{p}$ and negligible in the practical setting (e.g., $p > 2^{32}$).

Assuming that c_j is not a zero, we can notice that

$$a = 0 \Rightarrow \left(\frac{c_j}{p}\right) = \left(\frac{b_j r_j'^2}{p}\right) = b_j \text{ with prob. } 1, \text{ and that}$$

$$a \neq 0 \Rightarrow \left(\frac{c_j}{p}\right) = b_j \text{ with prob. } \frac{1}{2}.$$

The case of $a = 0$ is obvious. When $a \neq 0$, c_j is uniformly random whether b_j is -1 or 1 because ar_j is uniformly random, so the probability that $\left(\frac{c_j}{p}\right) = b_j$ is $\frac{1}{2}$.

Then the parties compute for $1 \leq j \leq k$,

$$[x_j]_p = \begin{cases} 2^{-1}([b_j]_p + 1) & \text{if } \left(\frac{c_j}{p}\right) = 1 \\ -2^{-1}([b_j]_p - 1) & \text{if } \left(\frac{c_j}{p}\right) = -1. \end{cases}$$

Note that $x_j \in \{0, 1\}$ and that $x_j = 1$ iff $\left(\frac{c_j}{p}\right) = b_j$. Finally, the parties compute $[a = 0]_p = [\wedge_{j=1}^{k} x_j]_p$ by using an unbounded fan-in And, assuming that at least one of x_j's is 0 if $a \neq 0$ with sufficiently large k.

The error probability that $(a = 0) = 1$ when $a \neq 0$ is $\left(\frac{1}{2}\right)^k$ and it can be negligible if we use sufficiently large k.

Similarly, when $p = 5 \bmod 8$, the parties compute and reveal for $1 \leq j \leq k$

$$c_j = ar_j + b_j' r_j'^2 \bmod p$$

instead of $c_j = ar_j + b_j r'^2_j \bmod p$ where $b'_j = -2^{-1}(b_j - 3)$.

Note that $b'_j \in_R \{2, 1\}$ because $b_j \in_R \{-1, 1\}$. Therefore, noting that $\left(\frac{2}{p}\right) = -1$, we can notice that

$$a = 0 \Rightarrow \left(\frac{c_j}{p}\right) = \left(\frac{b'_j r'^2_j}{p}\right) = b_j \text{ with prob. 1, and that}$$

$$a \neq 0 \Rightarrow \left(\frac{c_j}{p}\right) = b_j \text{ with prob. } \frac{1}{2}.$$

The rest of computation can be done as we did for $p = 3 \bmod 4$.

Though we assumed, for simplicity, that $p = 3 \bmod 4$ or that $p = 5 \bmod 8$, actually we can extend the idea to adapt to arbitrary primes if we generate $b_j \in_R \{y, 1\}$ such that $\left(\frac{y}{p}\right) = -1$.

3.6.5.2 Quadratic Residuosity Test Protocol

Incidentally, by using the random secret $b_j r_j^2$ in Sect. 3.6.5.1, we can also construct a quadratic residuosity test protocol where, given $[a \in \mathbb{Z}_p^*]_p$, the parties can compute $[\left(\frac{a}{p}\right)]_p$ as follows:

Here we assume that $p = 3 \bmod 4$ for simplicity. The parties generate $[br^2]_p$ in the same way as $b_j r_j^2$ is generated in Sect. 3.6.5.1, and reveal $c = br^2 a$. If c is a zero, the parties retry. The parties can compute $[\left(\frac{a}{p}\right)]_p$ as $\left(\frac{c}{p}\right)[b]_p$ because $\left(\frac{c}{p}\right) = \left(\frac{b}{p}\right)\left(\frac{a}{p}\right) = b\left(\frac{a}{p}\right)$.

3.6.5.3 Complexity of Probabilistic Equality Test Protocol

The complexity of computing each component is as follows: 3 rounds (including 2 rounds for random value generation) and $7k$ invocations for $[c_j]_p$'s and 1 rounds and $5k$ invocations for $[\wedge_{j=1}^{k} x_j]_p$. The total complexity is 4 rounds (including 2 rounds for random value generation) and $12k$ invocations.

3.6.5.4 Slight Improvement

We can utilize the theorem of [Per52] used in [Tof07] and slightly improve the efficiency of our probabilistic equality test protocol by computing $[c_j]_p$ as $c_j = a + b_j r'^2_j$ instead of $c_j = ar_j + b_j r'^2_j$. Therefore, we need $10k$ invocations instead of $12k$ in total.

This follows from the Perron's theorem below about the distribution property of quadratic residues. Based on this property, we still have that the probability that $\left(\frac{c_j}{p}\right) = b_j$ is roughly $\frac{1}{2}$ when $c_j = a + b_j r'^2_j$ and $a \neq 0$.

Theorem 2 ([Per52, Bau02]) *i. Let $p = 4k - 1$. Let $r_1, r_2, \ldots, r_{2k}$ be the $2k$ quadratic residues modulo p with 0 and let a be a non-zero number. Then among the $2k$ numbers $\{r_1 + a, \ldots, r_{2k} + a\}$, there are k quadratic residues (possibly including 0) and k quadratic nonresidues.*

ii. Let $p = 4k - 1$. Let $n_1, n_2, \ldots, n_{2k-1}$ be the $2k - 1$ quadratic nonresidues modulo p and let a be a non-zero number. Then among the $2k - 1$ numbers $\{n_1 + a, \ldots, n_{2k-1} + a\}$, there are k quadratic residues (possibly including 0) and $k - 1$ quadratic nonresidues.

iii. Let $p = 4k + 1$. Let $r_1, \ldots, r_{2k+1}$ be the $2k + 1$ quadratic residues modulo p with 0 and let a be a quadratic residue. Then among the $2k + 1$ numbers $\{r_1 + a, \ldots, r_{2k+1} + a\}$, there are $k + 1$ quadratic residues (including 0) and k quadratic nonresidues. If a is a quadratic nonresidue, there are k quadratic residues (not including 0) and $k + 1$ quadratic nonresidues.

iv. Let $p = 4k + 1$. Let $n_1, \ldots, n_{2k}$ be the $2k$ quadratic residues modulo p and let a be a quadratic residue. Then among the $2k$ numbers $\{n_1 + a, \ldots, n_{2k} + a\}$, there are k quadratic residues (not including 0) and k quadratic nonresidues. If a is a quadratic nonresidue, there are $k + 1$ quadratic residues (including 0) and $k - 1$ quadratic nonresidues.

3.6.6 Application of Equality Test Protocol

We consider private multiparty look-up tables (mLUT) [AC06, FGM07] as an application of our equality test protocols and show how our efficient equality test protocols can contribute to providing one possible realization of private mLUT.

The functionality of private mLUT is as follows: parties have an array of shared secrets $([a_1]_p, [a_2]_p, \ldots, [a_k]_p)$ and a shared index $[\sigma]_p$ and compute the output sharing $[a_\sigma]_p$.

The private mLUT is an important building block for secure protocols of private distributed constraint satisfaction problems [NZ05] and private stable matching problems [Gol06, FGM07b]. In [FGM07], the technique called *multiparty oblivious transfer* was developed, which is a generalization of Naor-Nissim indirect indexing [NN01]. On the other hand, in [AC06], the equality

test protocol is used and the output sharing $[a_\sigma]_p$ can be computed as

$$\sum_{i=1}^{k} [i = \sigma]_p \times [a_i]_p$$

where our equality test protocols can be used because of the attractive property that the outputs are also shared. In this case, the equality test protocol can be executed in parallel, so our round-efficient probabilistic and deterministic protocols can realize round-efficient private mLUT.

3.7 Implementation

In the real implementation, we can use (odd-even) parallel prefix computation [LF80, JA03] based on carry propagation and generation for the bitwise less-than and bitwise sum protocols as in [BDJ+06, FJ06, Tof05] where the complexity of bitwise less-than is roughly $2 + \log_2(\ell)$ rounds and $3\ell - 1$ invocations ($2\ell - 1$ invocations if one of the two operands is known) and the complexity of bitwise sum is roughly $2\log_2(\ell) - 1$ rounds and $5\ell - 2\log_2(\ell) - 4$ invocations ($4\ell - 2\log_2(\ell) - 4$ invocations if one of the two operands is known). Also, instead of joint random number sharing, we can use non-interactive pseudo-random secret sharing by Cramer, Damgård and Ishai [CDI05] in the secret sharing setting in order to reduce the round and communication complexities. In Table 3.3, we summarize the number of invocations of main subprotocols in each protocol. Whether we use constant-round subprotocols or non-constant-round subprotocols as building blocks, our constructions are more efficient according to Table 3.3. Though, in the comparison protocol, we need 3 invocations of joint random number bitwise-sharing compared with 2 in [DFK+06], this can be done in advance and our protocol seems more advantageous in real applications.

Table. 3.3 Number of Invocations of Subprotocols

Protocol		Random Bitwise-Sharing	Bitwise Less-Than	Bitwise Sum
Bit-Decomposition	[DFK+06]	1	1	2
	Proposed	1	1	1
Interval Test	[DFK+06]	1	3	2
	Proposed	1	2	0
Comparison	[DFK+06]	2	3	4
	Proposed	3	3	0
Equality Test	[DFK+06]	1	1	2
	Proposed1	1	0	0

Chapter 4

Multiparty Computation for Distributed Key Generation of Paillier Cryptosystem

In this chapter, we focus on the distributed key generation (DKG) protocol for the Paillier cryptosystem [Pai99]. The threshold Paillier cryptosystem is a typical threshold homomorphic cryptosystem (THC) used for multiparty computation (MPC). A special type of RSA modulus ([RSA78]) is necessary for the threshold Paillier cryptosystem and often such an RSA modulus is assumed to be given by a trusted dealer (i.e., authority) that also distributes the shares of the decryption key.

In order to remove such a trusted dealer, again we can make use of multiparty computation based on secret sharing. There are several known techniques used for generating an RSA modulus in a distributed way and we show how such techniques can be combined and adapted to the setting of the threshold Paillier cryptosystem without a trusted dealer. The results in this chapter were unpublished results due to the author.

4.1 Introduction

Many MPC protocols using THCs are Paillier-based construnctions (e.g., [CDN01, BFP+01, DN03, HN05, HN06, ST06, Gol06, FGM07b]). Therefore, the DKG protocol for the threshold Paillier cryptosystem is one of the most important applications of MPC because we can remove the trusted dealer and it is the main purpose of MPC.

Related Work. The DKG protocol (e.g., [Ped91]) for discrete-log based cryptosystems (e.g., [ElG85]) is relatively easy and simple. However, the DKG protocol for RSA-based cryptosystems is a non-trivial task. Boneh and Franklin [BF97] proposed the first DKG protocol for the RSA cryptosystem. Unfortunately it cannot be applied to the threshold RSA cryptosystem [Sho00] because for a technical reason the RSA modulus for [Sho00] must be a special type of RSA modulus that is a product of two large safe primes where a prime $p(= 2p' + 1)$ is a safe prime if p' is also a prime. For the same technical reason, the threshold Paillier cryptosystem [FPS00] also needs an RSA modulus that is a product of two safe primes. Damgård and Koprowski [DK01] proposed how to use an RSA modulus that is not a product of two safe primes in order to relax the condition, but the security proofs of their scheme were based on the non-standard complexity assumption. On the other hand, Fouque and Stern [FS01] proposed how to use an RSA modulus that is not a product of two safe primes and the security proofs of their scheme were based solely on the standard complexity assumption. Algesheimer, Camenisch, and Shoup [ACS02] proposed a novel DKG protocol that can generate such a special type of RSA modulus directly. However, finding safe primes can be very time-consuming and we are not aware whether there are infinitely many safe primes, so it will be more flexible for us to be able to use a wider class of RSA moduli as in [FS01]. Damgård and Jurik proposed a generalized Paillier cryptosystem [DJ01] and another homomorphic cryptosystem [DJ03] that can be considered as a mix of ElGamal and Paillier cryptosystems and both the threshold versions of [DJ01, DJ03] need RSA moduli that are products of two safe primes. One advantage of [DJ03] over the Paillier cryptosystem is that because the secret key of [DJ03] can be generated at random without the knowledge of the factors of the RSA modulus, the RSA modulus can be system-wide and reused by any set of parties that performs

MPC. Our distributed sieving protocol is applicable to both [DJ01] and [DJ03]. Damgård and Dupont [DD05] proposed how to use general RSA moduli for threshold RSA signatures with the observation that resultant signatures can be verified with the public key after they are generated from signature shares even if an adversary has a non-negligible chance of giving acceptable proofs for bad signature shares. However, the technique in [DD05] requires that we can recognize that the threshold signature generation or threshold decryption is done correctly and the techinique will not be applied to the context of our threshold decryption in the worst case because we may not be able to recognize the correct subset of decryption shares in case of $t < \frac{n}{2}$ (rather than $t < \frac{n}{3}$) where n is the number of parties and t is the number of parties the adversary can corrupt. Also, our distributed sieving protocol will still be useful for [DD05] because it can guarantee that $\frac{p-1}{2}$ and $\frac{q-1}{2}$ do not contain small factors where $N = pq$ is the RSA modulus and identify faulty parties by making the error probability of zero knowledge proofs small.

Building on [BF97, FS01, ACS02], we show how the techniques from [BF97, FS01, ACS02] can be utilized to realize the threshold Paillier cryptosystem as well as the threshold RSA signature scheme without a trusted dealer.

4.2 Building Blocks

In addition to the building blocks introduced in Chapter 3, we make use of the following.

4.2.1 Original Paillier Cryptosystem

First we describe the original Paillier cryptosystem, which is based on the *Decisional Composite Residuosity Assumption* (DCRA). That is, the semantic security of this cryptosystem is based on the difficulty of distinguishing N-th residues from non-N-th residues. The encryption and decryption processes are as follows.

- $N = pq$ (RSA modulus). We assume the bit length of p is the same as that of q.
- The public key PK is a pair of (N, g) where $g = N + 1$.
- The decryption key SK is $\lambda(N) = \mathrm{lcm}(p-1, q-1)$ where λ is called a Carmichael lambda function.

- Encryption:

  Given a plaintext $m \in \mathbb{Z}_N$, the ciphertext c is

  $$c = E(m, x) \equiv g^m x^N \bmod N^2$$

  where $x \in \mathbb{Z}_N^*$ is a random number.

  Note that

  $$E(m_1, x_1) \times E(m_2, x_2) = E(m_1 + m_2, x_1 x_2) \bmod N^2.$$

- Decryption:

  $$m = D(c) \equiv \frac{L(c^{\lambda(N)} \bmod N^2)}{L(g^{\lambda(N)} \bmod N^2)} \bmod N$$

  where $L(u) = \frac{u-1}{N}$, $u \in \{u < N^2 \mid u \equiv 1 \bmod N\}$.

  Note that L-function is not a modular computation.

Proof of correctness: From the definition of Carmichael λ function,

$$\lambda(N^2) = \text{lcm}(\varphi(p^2),\ \varphi(q^2)) = \text{lcm}(p(p-1),\ q(q-1)) = N\lambda(N),$$

where φ is the Euler φ function.

Because of the property of Carmichael λ function, we have

$$^\forall x \in \mathbb{Z}_{N^2}^*,\ \ x^{\lambda(N^2)} = x^{N\lambda(N)} \equiv 1 \bmod N^2.$$

Therefore, $c^{\lambda(N)} = (g^m x^N)^{\lambda(N)} = g^{m\lambda(N)} x^{N\lambda(N)} \equiv g^{m\lambda(N)} \bmod N^2$.

As a result, we have

$$\begin{aligned}
D(c) &\equiv \frac{L(c^{\lambda(N)} \bmod N^2)}{L(g^{\lambda(N)} \bmod N^2)} \\
&\equiv \frac{L(g^{m\lambda(N)} \bmod N^2)}{L(g^{\lambda(N)} \bmod N^2)} \\
&\equiv \frac{L(1 + \lambda(N)mN \bmod N^2)}{L(1 + \lambda(N)N \bmod N^2)} \quad (\because g = N + 1) \\
&\equiv \frac{\lambda(N)m \bmod N}{\lambda(N) \bmod N} \\
&\equiv m \bmod N.
\end{aligned}$$

□

Actually, we can use any g if the order of g is a multiple of N in $\mathbb{Z}_{N^2}$. Here we can use the special $g = N + 1$ whose order is exactly N in $\mathbb{Z}_{N^2}$ without degradation of security [DJ01].

4.2.2 Threshold Paillier Cryptosystem

The threshold Paillier cryptosystem was proposed in [FPS00], which uses the technique similar to the threshold RSA signature scheme [Sho00]. The decryption scheme is a bit different from that of the original Paillier cryptosystem. We describe the $(t + 1, n)$ threshold Paillier cryptosystem [FPS00] with a trusted dealer.

Key Generation and Distribution Phase:

1. The trusted dealer generates an RSA modulus $N = pq$ where p and q are safe primes. That is, $p = 2p' + 1$, and $q = 2q' + 1$ where p' and q' are also prime, and $\gcd(N, \varphi(N)) = 1$.
2. The dealer picks up $\beta \in \mathbb{Z}_N^*$ at random, and computes the following values,

   $$m = p'q', \quad \theta = m\beta \bmod N, \quad \Delta = n!.$$

   Because $\lambda(N^2) = \mathrm{lcm}(\varphi(p^2), \varphi(q^2)) = 2Nm$, note that

   $$^{\forall} x \in \mathbb{Z}_{N^2}^*, \quad x^{2Nm} \equiv 1 \bmod N^2.$$

3. The public key PK and the decryption key SK are $PK = (N, \ g, \ \theta)$ and $SK = \beta m$ where $g = N + 1$.

   The ciphertext c of a message M is defined as $c = g^M x^N \bmod N^2$ where $x \in \mathbb{Z}_N^*$ is random.
4. In order to share the decryption key $SK = \beta m$, the dealer generates a polynomial f with a_i chosen at random from $\{0, 1, \ldots, Nm - 1\}$,

   $$f(x) = \beta m + a_1 x + a_2 x^2 + \cdots + a_t x^t \bmod Nm,$$

   and sends $f(i)$ to each party P_i $(1 \le i \le n)$ through the secure channel.

   Also the dealer picks up a random value $r \in \mathbb{Z}_{N^2}^*$ and publishes the verification keys VK and VK_is as

   $$VK = v = r^2 \bmod N^2,$$

$$VK_i = v^{\Delta f(i)} \bmod N^2.$$

These verification keys are necessary for the parties to prove that the decryption procedure is done correctly. For a technical reason, $v^{\Delta f(i)}$ is used instead of $v^{f(i)}$ for VK_i.

Decryption Phase:

Now the parties decrypt a ciphertext $c = g^M x^N \bmod N^2$ where M is the plaintext.

1. Each party P_i publishes $c_i = c^{2\Delta f(i)} \bmod N^2$ by using its secret share, which we call the partial decryption. Also P_i publishes the zero knowledge proof for $f(i) = \log_{v^\Delta} VK_i = \log_{c^{4\Delta}}(c_i)^2$. We accept only the partial decryptions with the valid zero knowledge proofs. The reason we use $f(i) = \log_{c^{4\Delta}}(c_i)^2$ instead of $f(i) = \log_{c^{2\Delta}} c_i$ is to make sure that we are working in Q_{N^2} as in [Sho00]. The structure of Q_{N^2} is explained in Sect. 4.2.2.1.
2. By combining $t + 1$ valid partial decryptions of the subset S of the parties, we can obtain

$$M = L(\prod_{i \in S} c_i^{2\mu_i} \bmod N^2) \times \frac{1}{4\Delta^2 \theta} \bmod N$$

where $\mu_i = \Delta \times \lambda_{0,i}^S \in \mathbb{Z}$, and $L(u) = \frac{u-1}{N}$.

Proof of correctness:

$$\begin{aligned}
\prod_{i \in S} c_i^{2\mu_i} &= c^{4\Delta \sum_{i \in S} f(i)\mu_i} \\
&= c^{4\Delta \sum_{i \in S} \Delta f(i)\lambda_{0,i}^S} \\
&= c^{4\Delta^2 m\beta} \\
&= (g^M x^N)^{4\Delta^2 m\beta} \\
&= g^{4\Delta^2 m\beta M} \quad (\because {}^\forall x,\ x^{2Nm} = 1 \bmod N^2) \\
&= 1 + 4\Delta^2 m\beta MN \bmod N^2 \quad (\because g = N + 1).
\end{aligned}$$

Therefore,

$$L(\prod_{i \in S} c_i^{2\mu_i} \bmod N^2) = 4\Delta^2 m\beta M = M \times 4\Delta^2 \theta \bmod N.$$

Δ and θ are public information. Thus, we have

$$M = L(\prod_{i \in S} c_i^{2\mu_i} \bmod N^2) \times \frac{1}{4\Delta^2\theta} \bmod N.$$

□

Note that anyone can verify that the dealer published the correct VK_i's as follows. By combining $t+1$ VK_i's of any subset S, we can obtain

$$\prod_{i \in S} VK_i^{\mu_i} = \prod_{i \in S} v^{\Delta^2 f(i)\lambda_{0,i}^S} = v^{\Delta^2(m\beta + kNm)} = r^{2\Delta^2(m\beta+kNm)} = r^{2\Delta^2 m\beta} \bmod N^2$$

for some integer k.

Therefore, by checking if $(r^{2\Delta^2 m\beta})^N \stackrel{?}{=} 1 \bmod N^2$, we can verify the correctness of VK_i's.

4.2.2.1 Why we need two safe primes

In this threshold scheme, the dealer has to generate two safe primes. The reason can be explained as follows similarly as in [Sho00]. We denote by Q_{N^2} the subgroup of squares in $\mathbb{Z}_{N^2}^*$. That is,

$$Q_{N^2} = \{x \in \mathbb{Z}_{N^2}^* \mid \left(\frac{x}{p}\right) = 1 \ \& \ \left(\frac{x}{q}\right) = 1\}$$

where $\left(\frac{x}{p}\right)$ is the Legendre symbol. By the Chinese Remainder Theorem,

$$|Q_{N^2}| = \frac{p-1}{2} \times p \times \frac{q-1}{2} \times q = Nm.$$

Also, Q_{N^2} is cyclic. Actually, by choosing $G_{Q_{N^2}}$ such that

$$G_{Q_{N^2}} \equiv g_{p^2}^2 \bmod p^2 \ \text{(where } g_{p^2} \text{ is a generator of } \mathbb{Z}_{p^2}^*)$$

$$G_{Q_{N^2}} \equiv g_{q^2}^2 \bmod q^2 \ \text{(where } g_{q^2} \text{ is a generator of } \mathbb{Z}_{q^2}^*),$$

the order of $G_{Q_{N^2}}$ is Nm, and generates Q_{N^2}.

Therefore, the number of the generators of Q_{N^2} is

$$\varphi(Nm) = \varphi(pqp'q') = (p-1)(q-1)(p'-1)(q'-1).$$

When the dealer picks up a random value $r \in \mathbb{Z}_{N^2}$, and computes $v = r^2 \bmod N^2$, v is a generator of Q_{N^2} with overwhelming probability $\frac{\varphi(Nm)}{Nm}$, and then $VK_i = v^{\Delta f(i)} \bmod N^2$ completely determines

$f(i) \bmod Nm$. Thus, if the order of $G_{Q_{N^2}}$ does not contain small factors, the probability that the party P_i can publish the bad partial decryption with a valid zero knowledge proof by finding $f'(i)$ such that $VK_i = v^{\Delta f(i)} = v^{\Delta f'(i)}$ and $f(i) \neq f'(i) \bmod Nm$ is negligible. This condition is important to prevent the parties from disrupting the decryption phase and to identify the faulty parties. On the other hand, if the order of $G_{Q_{N^2}}$ contains a small factor θ and it happens that $v^\theta = 1 \bmod N^2$, then $VK_i = v^{\Delta f(i)} = v^{\Delta(f(i)+k\theta)}$ where k is some integer. This means that instead of $f(i)$, $f(i) + k\theta$ can be used for publishing the bad partial decryption. That is why we need two safe primes to guarantee that the order of $G_{Q_{N^2}}$ does not contain small factors and to make the error probability of zero knowledge proofs small.

4.2.3 Secret Sharing over the Integers

In Sect. 3.3.1.1 of Chapter 3, we introduced a polynomial secret sharing scheme over a prime finite field. Here we need a secret sharing scheme over the integers [GJKR96, FGMY97, Rab98], which is a variant of [Sha79]. Typically this scheme is used in the setting where the modulus is also a (shared) secret. We describe the $(t+1, n)$ secret sharing over the integers.

Suppose the dealer wants to share a secret $s \in [-M, M]$. Then the dealer of the secret generates a polynomial,

$$f(x) = \Delta s + a_1 x + a_2 x^2 + \cdots + a_t x^t$$

where $\Delta = n!$, random values $a_i \in_R [-K\Delta^2 M, K\Delta^2 M]$ and $K = 2^{O(\kappa)}$ with security parameter κ chosen such that $1/K$ is negligible.

Now we assume that the adversary has collected t secret shares of a subset $S' \subset \{1, 2, \ldots, n\}$. We prove that with high probability a polynomial $r(x)$ exists such that $r(0) = \Delta s' (\neq \Delta s)$, $r(i) = f(i)$ for $i \in S'$, and the coefficients of $r(x)$ are in the correct range. If $r(x)$ exists for each guess s', then the adversary cannot guess the correct secret.

We consider a polynomial $h(x)$ such that $h(0) = \Delta(s - s')$ and $h(i) = 0$ for $i \in S'$. $h(x)$ can be computed by Lagrange interpolation as

$$h(x) = \sum_{i \in S' \cup \{0\}} h(i) \lambda_{x,i}^{S' \cup \{0\}}.$$

Therefore, if we say $S' = \{i_1, i_2, \ldots, i_t\}$

$$h(x) = \Delta(s - s')\frac{(x - i_1)(x - i_2)\cdots(x - i_t)}{(0 - i_1)(0 - i_2)\cdots(0 - i_t)}.$$

The coefficient of x^{t-i} of $h(x)$ is

$$\frac{\Delta(s - s')}{\prod_{j \in S'}(-j)} \times \sum_{B \subset S', |B| = i} \left(\prod_{j \in B} (-j) \right).$$

Because $\prod_{j \in S'}(-j)$ divides Δ, the coefficient of x^{t-i} is an integer. The coefficient is bounded (in absolute value)

$$\sum_{B \subset S', |B| = i} \Delta(s - s') \leq \Delta(s - s')\binom{t}{i} = \frac{\Delta(s - s') \times t!}{i! \times (t - i)!} \leq \Delta^2(s - s') \leq 2\Delta^2 M.$$

Note that if we say $r(x) = f(x) - h(x)$, then $r(0) = \Delta s'$, $r(i) = f(i)$ for $i \in S'$, the coefficients of $r(x)$ are integers, and in the range $[-(K\Delta^2 M + 2\Delta^2 M), K\Delta^2 M + 2\Delta^2 M]$. The probability that the coefficients of $r(x)$ will not be in the correct range is upper bounded by a union bound,

$$t \times \frac{2\Delta^2 M}{K\Delta^2 M} = \frac{2t}{K}$$

which is negligible if K is sufficiently huge (say $K = 2^{100}$) because t is typically small. Therefore, for each guess s', with high probability, there exists a valid polynomial $r(x)$. Thus, the adversary cannot guess the correct secret. The proof here is a variant of the proof in [Rab98].

We usually use the following modified polynomial,

$$f(x) = \Delta(\Delta s + a_1 x + a_2 x^2 + \cdots + a_t x^t)$$

where $\Delta = n!$ and $a_i \in_R [-K\Delta^2 M, K\Delta^2 M]$.

The advantage of using this polynomial is that all the shares $f(i)$'s become the multiples of Δ. When we reconstruct $\Delta^2 s$ by

$$\Delta^2 s = f(0) = \sum_{i \in S} f(i)\lambda_{0,i}^S,$$

$f(i)\lambda_{0,i}^S$ is an integer because $f(i)$ is a multiple of Δ and the denominator of $\lambda_{0,i}^S$ divides Δ. Therefore, the intermediate computation can be done over the integers, not including the rational numbers. Note that the reconstructed secret we have when a secret integer s is shared over the integers is $\Delta^2 s$ instead of s.

4.2.4 BGW Protocol Modulo a Non Prime

In Sect. 3.3.2 in Chapter 3, we assumed that all arithmetic operations were performed modulo a prime. However, it is also possible for the parties to run the BGW protocol modulo a non prime M as mentioned in [BF97]. If M has no prime divisors smaller than n where n is the number of all the parties, then the BGW protocol can be used in the same way as in Sect. 3.3.2 because the denominators of the Lagrange coefficients can have the inverses modulo M.

Running the BGW protocol modulo M containing small factors needs a slight modification. In our setting considered in Sect. 4.4.1, we have only to consider M that is a prime smaller than n. For example, let's consider $M = 3$. If $n \geq M(= 3)$, we cannot use Shamir secret sharing modulo M because we cannot have enough points in the filed $\mathbb{F}_M$. However, we can handle such a case by using Shamir secret sharing over an algebraic extension field of $\mathbb{F}_M$ that can contain more than n points.

4.2.5 Joint Random Invertible Number Sharing

This protocol is due to [BB89]. By using the joint random number sharing in Sect. 3.3.5.1, the parties generate two random sharings $[s]_p$ and $[r]_p$, compute $[sr]_p$ and reveal sr. If $sr \neq 0$, then the parties can obtain the sharing $[r]_p$ where r is a random invertible number. Otherwise, the parties repeat the protocol. This protocol can also be used over the extension field $\mathbb{F}_{p^e}$.

If the computation is done modulo a non prime M, the condition $sr \neq 0$ is replaced with $\gcd(sr, M) = 1$ and if this condition holds, the parties can obtain a sharing $[r]_M$ where r is an invertible element in the ring $\mathbb{Z}_M$.

Alternatively, each party P_i can pick up an invertible element $r_i \in \mathbb{Z}_M^*$ (or $r_i \in \mathbb{F}_p^*$) and compute $[r]_M = \prod_{i=1}^n [r_i]_M$ (or $[r]_{\mathbb{F}_{p^e}} = \prod_{i=1}^n [r_i]_{\mathbb{F}_{p^e}}$).

4.3 DKG Protocol [BF97]

The protocols [BF97, FMY98, MWB99] describe how to generate an RSA modulus and share the secret key in a distributed way. [MWB99] describes the implementation and optimization of

[BF97] in detail. [FMY98] has added robustness to [BF97] in order to cope with any minority of malicious parties. For simplicity, we focus on [BF97] and its RSA modulus generation part and we do not inlcude the optimization used in [MWB99, FS01] in our description though it can also be used for us. We also need these protocols to construct our threshold Paillier cryptosystem without a trusted dealer.

4.3.1 High-Level Overview

In [BF97], the parties perform the following basic steps to compute an RSA modulus $N = pq$.

1. Every party P_i except P_1 picks up random $(k - \log_2(n) - 1)$-bit secrets p_i and q_i such that $p_i \equiv q_i \equiv 0 \bmod 4$. P_1 also picks up random $(k - \log_2(n) - 1)$-bit secrets p'_1 and q'_1 such that $p'_i \equiv q'_i \equiv 3 \bmod 4$ and sets $p_1 = 2^{k-1} + p'_1, q_1 = 2^{k-1} + q'_1$. Here n is the number of the parties. Clearly, shared candidates $p = \sum_{i=1}^{n} p_i$ and $q = \sum_{i=1}^{n} q_i$ are k-bit integers. The distributions of p and q are not uniform but as shown in [BF97], the distributions have at least $(k - \log_2(n) - 1)$-bits of entropy. We assume that n is odd and $n = 2\ell + 1$ for some ℓ. The protocol here is designed to be ℓ private. That is, any subset of ℓ or less than ℓ parties cannot reveal the factorization of N.
2. The parties agree on some large prime P where $P > 2^{2k} > N$, and all arithmetic operations are done modulo P.
3. By using the BGW protocol [BGW88], the n parties compute

$$N = pq = (p_1 + \cdots + p_n) \times (q_1 + \cdots + q_n) = \sum_{i=1}^{n} p_i \times \sum_{i=1}^{n} q_i \bmod P$$

   without revealing p and q. Because $N < P$, the parties can compute N.
4. The parties perform biprimality test for checking if N is a product of two primes in a distributed way. If the test fails, the protocol is restarted.

Note that after the computation is done, the parties have additive shares of p and q over the integers.

4.3.2 Distributed Computation of RSA Modulus N by BGW Protocol

We see how the parties compute and publish $N = (\sum p_i) \times (\sum q_i)$ by using the classical BGW protocol in detail.

1. Each party P_i generates two random ℓ-degree polynomials $f_i, g_i \in \mathbb{Z}_P[x]$ such that

$$f_i(x) = p_i + a_1 x + a_2 x^2 + \cdots + a_\ell x^\ell \bmod P,$$

$$g_i(x) = q_i + b_1 x + b_2 x^2 + \cdots + b_\ell x^\ell \bmod P.$$

   Also, P_i generates a random 2ℓ-degree polynomial $h_i \in \mathbb{Z}_P[x]$ such that

$$h_i(x) = 0 + c_1 x + c_2 x^2 + \cdots + c_{2\ell} x^{2\ell} \bmod P.$$

   Note that $f_i(0) = p_i$, $g_i(0) = q_i$, and $h_i(0) = 0$. $h_i(x)$ is necessary for randomization in the BGW protocol.

2. P_i computes the following values:

$$p_{i,j} = f_i(j), \;\; q_{i,j} = g_i(j), \;\; h_{i,j} = h_i(j) \;\text{ for }\; j = 1, \ldots, n.$$

   P_i sends the triple $\langle p_{i,j}, \;\; q_{i,j}, \;\; h_{i,j} \rangle$ to P_j through the secure channel.

3. After receiving the above triple from other parties, P_i computes

$$N_i = \left(\sum_{j=1}^{n} p_{j,i} \right) \left(\sum_{j=1}^{n} q_{j,i} \right) + \sum_{j=1}^{n} h_{j,i} \bmod P,$$

   and P_i publishes N_i.

4. Now we consider the following polynimial,

$$\alpha(x) = \left(\sum_{j=1}^{n} f_j(x) \right) \left(\sum_{j=1}^{n} g_j(x) \right) + \sum_{j=1}^{n} h_j(x) \bmod P.$$

   Note that $N_i = \alpha(i)$, and $\alpha(0) = N$. Because $\alpha(x)$ is a 2ℓ-degree polynomial, and we have n $(= 2\ell + 1)$ shares, we can compute coefficients of $\alpha(x)$ by using Lagrange interpolation. Thus, we can obtain

$$N = \alpha(0) = \sum_{i \in S} N_i \lambda_{0,i}^S \bmod P$$

where $S = \{1, 2, \ldots, n\}$.

Note that if the parties do not publish N_i, the parties can have N in the additive sharing by computing $N_i\lambda_{0,i}^S$ locally without revealing N. That is, $N = \sum_{i=1}^{n} s_i$ where $s_i = N_i\lambda_{0,i}^S$ and s_i is a share for P_i. That is, we can convert a polynomial sharing into an additive sharing easily in this way. This technique is called *poly-to-sum* [FGMY97] and we will use this conversion in Sect. 4.5.

4.3.3 Distributed Biprimality Test for N

Boneh and Franklin showed how the parties can check if N is a product of two primes without revealing p and q. Their novel distributed biprimality test consists of two parts. Here we assume that $p \equiv q \equiv 3 \bmod 4$. This condition can be assured by having P_1 pick up the secrets $p_1 \equiv q_1 \equiv 3 \bmod 4$, and all other parties pick up the secrets $p_i \equiv q_i \equiv 0 \bmod 4$.

Biprimality Test 1:

1. The parties agree on a random $g \in \mathbb{Z}_N^*$ such that the Jacobi symbol $\left(\frac{g}{N}\right) = 1$.
2. P_1 broadcasts $v_1 \equiv g^{\frac{N-p_1-q_1+1}{4}} \bmod N$. All other parties broadcast $v_i \equiv g^{-\frac{p_i+q_i}{4}} \bmod N$.
3. The parties compute and check if

$$v \equiv \prod_{i=1}^{n} v_i \equiv g^{\frac{N-p-q+1}{4}} \stackrel{?}{\equiv} \pm 1 \bmod N.$$

If $v \not\equiv \pm 1$, the parties declare that N is not a product of two primes.

Note that if N is a product of two primes, then

$$v \equiv g^{\frac{N-p-q+1}{4}} \equiv g^{\frac{(p-1)(q-1)}{4}} \equiv g^{\frac{\varphi(N)}{4}} \bmod N.$$

Sketch of proof: Suppose p and q are distinct primes. Then,

$$\left(\frac{g}{p}\right) = \left(\frac{g}{q}\right) \quad (\because \left(\frac{g}{N}\right) = 1).$$

Because $\frac{p-1}{2}$ and $\frac{q-1}{2}$ are odd ($\because p \equiv q \equiv 3 \bmod 4$), we have,

$$g^{\frac{\varphi(N)}{4}} \equiv (g^{\frac{p-1}{2}})^{\frac{q-1}{2}} \equiv \left(\frac{g}{p}\right)^{\frac{q-1}{2}} \equiv \left(\frac{g}{p}\right) \bmod p,$$

$$g^{\frac{\varphi(N)}{4}} \equiv (g^{\frac{q-1}{2}})^{\frac{p-1}{2}} \equiv \left(\frac{g}{q}\right)^{\frac{p-1}{2}} \equiv \left(\frac{g}{q}\right) \bmod q.$$

Thus, it follows that $g^{\frac{\varphi(N)}{4}} \equiv \pm 1 \bmod N$. Therefore, if N is a product of two primes, N is always accepted by the parties.

When N is not a product of two primes, we consider two subgroups G and H of $\mathbb{Z}_N^*$ such that

$$G = \{x \mid \left(\frac{x}{N}\right) = 1\},$$

$$H = \{x \mid x \in G \ \& \ x^{\frac{N-p-q+1}{4}} \equiv \pm 1 \bmod N\}.$$

Note that H is a subgroup of G. If N is a product of two primes, we know $H = G$ as we have seen above. If N is not a product of two primes, we can prove $|H| \leq \frac{|G|}{2}$ by showing the existence of an element x such that $x \in G \ \& \ x \notin H$. The more details can be found in [BF97]. Therefore, we have at least $\frac{1}{2}$ witnesses in G to reject N which is not a product of two primes. We can repeat the biprimality test1 to obtain the desired security level. □

Biprimality Test 2:

The parties perform the additional biprimality test2 to cope with the special N where

$$N = pq, \ \ p = r_1^{d_1}, \ \ q = r_2^{d_2}, \ \ d_1 > 1, \ \ q \equiv 1 \bmod r_1^{d_1-1},$$

and r_1 and r_2 are primes.

In this special case, it may happen that $H = G$, and the parties may accept N falsely though N is not a product of two primes. In order to reject such a special N, we consider a group $\mathbb{T}_N$ and its subgroup H' such that

$$\mathbb{T}_N = (\mathbb{Z}_N[x]/(x^2+1))^*/\mathbb{Z}_N^*,$$

$$H' = \{x \mid x \in \mathbb{T}_N \ \& \ x^{(p+1)(q+1)} = 1\}.$$

The parties perform the Fermat test in $\mathbb{T}_N$ as follows.

1. The parties agree on a random $h \in \mathbb{T}_N$.
2. P_1 broadcasts $u_1 = h^{N+p_1+q_1+1}$. All other parties broadcast $u_i = h^{p_i+q_i}$.
3. The parties compute and check if

$$u = \prod_{i=1}^{n} u_i = h^{N+p+q+1} = h^{(p+1)(q+1)} \stackrel{?}{=} 1.$$

If $u \neq 1$, the parties declare that N is not a product of two primes.

Sketch of proof: Similarly, suppose p and q are distinct primes. Then the polynomial $x^2 + 1$ has no root in $\mathbb{F}_p$ and $\mathbb{F}_q$ because $p \equiv q \equiv 3 \bmod 4$. Therefore, $\mathbb{F}_p[x]/(x^2 + 1)$ and $\mathbb{F}_q[x]/(x^2 + 1)$ are quadratic extensions of $\mathbb{F}_p$ and $\mathbb{F}_q$. It follows that the order of $\mathbb{T}_p = (\mathbb{Z}_p[x]/(x^2 + 1))^*/\mathbb{Z}_p^*$ is $p + 1$. This can be confirmed by

$$|\mathbb{T}_p| = \frac{|(\mathbb{Z}_p[x]/(x^2 + 1))^*|}{|\mathbb{Z}_p^*|} = \frac{p^2 - 1}{p - 1} = p + 1.$$

Similary, $|\mathbb{T}_q| = q + 1$. By the Chinese Remainder Theorem, $\mathbb{T}_N = \mathbb{T}_p \times \mathbb{T}_q$. Hence

$$|\mathbb{T}_N| = |\mathbb{T}_p| \times |\mathbb{T}_q| = (p + 1)(q + 1).$$

Thus,

$$^{\forall} h \in \mathbb{T}_N, \;\; h^{|\mathbb{T}_N|} = h^{(p+1)(q+1)} = 1.$$

Therefore, if N is a product of two primes, N is always accepted by the parties.

If N is not a product of two primes, we can prove $|H'| \leq \frac{|\mathbb{T}_N|}{2}$ by showing the existence of an element x such that $x \in \mathbb{T}_N \;\&\; x \notin H'$. The more details can be found in [BF97]. Therefore, we have at least $\frac{1}{2}$ witnesses in $\mathbb{T}_N$ to reject N which is not a product of two primes. We can repeat the biprimality test2 to obtain the desired security level. □

4.4 Relaxing Condition on Safe Primes

As we have seen in Sect. 4.2.2, the dealer needs two safe primes to generate an RSA modulus $N = pq$ for the threshold Paillier cryptosystem. That is, $p = 2p' + 1$, $q = 2q' + 1$ where p', and q' are also primes. The reason we need safe primes is to make Q_{N^2} a cyclic group and have

the sufficiently large number of generators in Q_{N^2}. By choosing a generator for the verification key VK, each VK_i can completely determine the party's secret share, and prevent the party from publishing the false decryption. Furthermore, if N is a product of two safe primes, an element chosen at random from Q_{N^2} is a generator with overwhelming probability.

However, as mentioned in [FS01], we can observe that even if p' and q' are not primes, Q_{N^2} can be cyclic if $\gcd(p', q') = 1$. Also, by making sure that p' and q' do not contain the prime factors that are not more than a sufficiently large sieving bound B, we can ensure that the number of the generators in Q_{N^2} is large.

We show how these conditions can be assured by our improvements to the protocols in [FS01].

4.4.1 Our Improved Distributed Sieving Protocol for p' and q'

The basic idea behind [FS01] is to make sure that

$$\gcd(p - 1, 4B') = \gcd(2p', 4B') = 2,$$

$$\gcd(q - 1, 4B') = \gcd(2q', 4B') = 2$$

where $B' = 3 \times 5 \times 7 \times 11 \times \cdots \times B$ (a product of all the primes up to a sieving bound B). When these conditions hold, p' and q' do not contain the prime factors up to B and it ensures that we have many generators in Q_{N^2}. In [FS01], it is suggested that $B > 2^{16}$.

In [FS01], the parties compute and publish

$$T = (p - 1) + 4B'R'$$

over the integers by using the simplified version of the GCD protocol in [CGS00] where R' is a random secret integer chosen appropriately. By computing $\gcd(T, 4B')$, the parties can check if $\gcd(p - 1, 4B') = 2$ holds. However, this leaks the information $p - 1 \bmod 4B' (= T \bmod 4B')$. Even if we compute $T = (p - 1)R + 4B'R'$ by using the GCD protocol where R and R' are random secret integers chosen appropriately, it can happen that $\gcd(T, 4B') \neq 2$ though $\gcd(p-1, 4B') = 2$ when $\gcd(R, 4B') \neq 1$. Then the good candidate p will be rejected. We avoid such a problem by combining the trial division protocol in [ACS02] and the BGW protocol modulo a non prime [BF97].

The parties perform the following steps for both p and q. Here we describe only the case of p. At the beginning, the parties have the shares such that $p = \sum_{i=1}^{n} p_i$.

Proposed Distributed Sieving Protocol for p' and q':

1. The parties can easily share
$$p' = \frac{p-1}{2} = \frac{p_1 - 1}{2} + \sum_{i=2}^{n} \frac{p_i}{2} = p'_1 + \sum_{i=2}^{n} p'_i = \sum_{i=1}^{n} p'_i.$$
where p'_i is each party's additive share over the integers.
2. Write $B' = M_0 \prod_{j=1}^{u} M_j$ where M_0 has no prime factors smaller than n and each M_j is a prime smaller than n.
3. In order to check if $\gcd(p', 2) = 1$, the parties can run the trial division protocol as follows. Note that actually $p'(= \frac{p-1}{2})$ is odd because $p \equiv 3 \bmod 4$, so the parties do not need to check if $\gcd(p', 2) = 1$. We describe the protocol here for completeness.
   (a) Each party P_i reshares its share p'_i over the appropriate extension field $\mathbb{F}_{2^v}$ by a polynomial sharing as in Sect. 3.3.1.1. Here we can assume that each party is assigned a unique point in the extension field $\mathbb{F}_{2^v}$ where $n < 2^v$.
   (b) The parties can share p' over $\mathbb{F}_{2^v}$ as in Sect. 3.3.2 by adding all the shares sent by other parties.
   (c) The parties reveal p', which is equal to $p' \bmod 2$ and if it is 0, it means that $\gcd(p', 2) = 2$ and the candidate p is discarded.
4. In order to check if $\gcd(p', M_0) = 1$, the parties run the trial division protocol as follows:
   (a) Each party P_i reshares its share p'_i modulo M_0 by a polynomial sharing as in Sect. 3.3.1.1. Here we can simply assume that each party P_i is assigned a point i.
   (b) The parties can share p' modulo M_0 as in Sect. 3.3.2 by adding all the shares sent by other parties.
   (c) The parties generate a random invertible number $[r]_{M_0}$ (see Sect. 4.2.5), compute $[rp']_{M_0}$ and reveal $rp' \bmod M_0$. If $\gcd(rp', M_0) \neq 1$, it means that p' contains some factor of M_0 and the candidate p is discarded.
5. In order to check if $\gcd(p', M_j) = 1$ for $1 \leq j \leq u$, the parties run the trial division protocol

as follows:

(a) Each party P_i reshares its share p'_i over the appropriate extension field $\mathbb{F}_{M_j^v}$ by a polynomial sharing as in Sect. 3.3.1.1. Here we can assume that each party is assigned a unique point in the extension field $\mathbb{F}_{M_j^v}$ where $n < M_j^v$.

(b) The parties can share p' over $\mathbb{F}_{M_j^v}$ as in Sect. 3.3.2 by adding all the shares sent by other parties.

(c) The parties generate a random invertible number $[r]_{\mathbb{F}_{M_j^v}}$ such that $r \in \mathbb{F}^*_{M_j}$ (see Sect. 4.2.5), compute $[rp']_{\mathbb{F}_{M_j^v}}$ and reveal rp', which is equal to $rp' \bmod M_j$. If $rp' = 0 \bmod M_j$, it means that p' contains factor M_j and the candidate p is discarded.

Note that the trial division protocol can be run in parallel to reduce the round complexity.

4.4.2 Making sure that $\gcd(p', q') = 1$

In order to make sure that $\gcd(p', q') = 1$, [FS01] uses the observation that

$$\gcd(p-1, q-1) | \gcd(N-1, \varphi(N)).$$

Proof: We have $\varphi(N) = (p-1)(q-1) = N - p - q + 1 = (N-1) - (p-1) - (q-1)$. Therefore, $(N-1) - \varphi(N) = (p-1) + (q-1)$. Thus, $\gcd(N-1,\ \varphi(N)) = \gcd((N-1) - \varphi(N),\ \varphi(N)) = \gcd((p-1) + (q-1),\ \varphi(N))$. Now we say $a = p-1,\ \ b = q-1$. Then,

$$\gcd(p-1, q-1) = \gcd(a, b),$$

$$\gcd(N-1, \varphi(N)) = \gcd(a+b, ab).$$

Obviously, $\gcd(a, b)$ divides $\gcd(a+b, ab)$. □

In [FS01], the parties compute and publish

$$T' = \varphi(N) + (N-1)R'$$

over the integers by using the simplified version of the GCD protocol in [CGS00] where R' is a random secret integer chosen appropriately. By computing $\gcd(T', N-1)$, the parties can check if

$\gcd(\varphi(N), N-1) = 4$ holds. However, this leaks the information $\varphi(N) \bmod N-1 (= T' \bmod N-1)$. Because $\varphi(N) < N-1$, the information $\varphi(N)$ is leaked. Even if we compute $T' = \varphi(N)R+(N-1)R'$ by using the GCD protocol where R and R' are random secret integers chosen appropriately, it can happen that $\gcd(T', N-1) \neq 4$ though $\gcd(\varphi(N), N-1) = 4$ when $\gcd(R, N-1) \neq 1$. Then the good candidate N will be rejected. We avoid such a problem as we did in Sect.4.4.1.

In our setting, we have $p \equiv 3 \bmod 4$ and $q \equiv 3 \bmod 4$, so we have $4|N-1$ and $4|\varphi(N) (= (p-1)(q-1))$. On the other hand, $\gcd(p-1, q-1) = \gcd(2p', 2q') = 2 \times \gcd(p', q')$. Therefore, if $\gcd(\frac{N-1}{4}, \frac{\varphi(N)}{4}) = 1$, we have $\gcd(p', q') = \gcd(\frac{p-1}{2}, \frac{q-1}{2}) = 1$.

Note that the parties have $\varphi(N)$ in the additive sharing over the integers after the computation of N. Actually,

$$\varphi(N) = N - p - q + 1 = \sum_{i=1}^{n} \varphi_i$$

where $\varphi_1 = N + 1 - p_1 - q_1$ and $\varphi_i = -p_i - q_i$ for $2 \leq i \leq n$. Each party P_i can compute φ_i locally. Furthermore, the parties have $\frac{\varphi(N)}{4}$ in the additive sharing over the integers because $4|\varphi_1$ and $4|\varphi_i$ for $2 \leq i \leq n$.

In order to check if $\gcd(\frac{N-1}{4}, \frac{\varphi(N)}{4}) = 1$ holds, we can take the same approach we proposed in Sect. 4.4.1. That is, in Sect. 4.4.1, B' is a public value and $p' = \frac{p-1}{2}$ is a shared secret and here $\frac{N-1}{4}$ is a public value and $\frac{\varphi(N)}{4}$ is a shared secret. Therefore, if $\frac{N-1}{4} = 2^{e_1} M_0 \prod_{j=1}^{u} M_j^{e_j}$ where M_0 has no prime factors smaller than n and each M_j is a prime smaller than n, the parties check if $\gcd(\frac{\varphi(N)}{4}, M_0) = 1$ and $\gcd(\frac{\varphi(N)}{4}, M_j) = 1$ for $1 \leq j \leq u$ hold by using our distributed sieving protocol in Sect. 4.4.1. Note that the parties do not need to check if $\gcd(\frac{\varphi(N)}{4}, 2) = 1$ because $\frac{\varphi(N)}{4} (= (2k_1 + 1)(2k_2 + 1))$ is odd where $p = 4k_1 + 3$ and $q = 4k_2 + 3$.

4.4.3 Generators of Q_{N^2}

When $\gcd(p', q') = 1$, and p' and q' do not contain the prime factors less than B, the number of the generators of Q_{N^2} is

$$\sharp(\text{generators of } Q_{N^2}) = \varphi(pqp'q') = pqp'q' \left(1 - \frac{1}{p}\right)\left(1 - \frac{1}{q}\right)\left(1 - \frac{1}{B_1}\right)\left(1 - \frac{1}{B_2}\right)\left(1 - \frac{1}{B_3}\right)\cdots$$

where B_i's are primes, and $B < B_1 < B_2 < B_3 < \cdots$.

By picking up k'-tuple verification keys $\langle VK_{\_1}, \ldots, VK_{\_k'} \rangle$ from Q_{N^2} at random, the probability

that the tuple of VK_i's generates Q_{N^2} is high if we adopt large enough security parameters k' and B as shown in [FS01].

4.5 Threshold Paillier Cryptosystem Without Trusted Dealer

We show how to construct the threshold Paillier cryptosystem without a trusted dealer building on the protocols introduced in this chapter. We revisit the threshold Paillier cryptosystem with a trusted dealer. As we have seen in Sect. 4.2.2, the dealer has to pick up a random $\beta \in \mathbb{Z}_N^*$ and compute $\theta = m\beta \bmod N$ where $m = p'q'$. Also, the dealer distributes the secret shares for βm by using the polynomial

$$f(x) = \beta m + a_1 x + a_2 x^2 + \cdots + a_t x^t \bmod Nm.$$

In order to remove the dealer, the parties must compute $\theta = m\beta \bmod N$ so that no parties can know β or m, and share βm over the integers instead of modNm because no parties should know m. We use the observations that $\varphi(N) = (p-1)(q-1) = 4m$ is a multiple of m, and the parties have $\varphi(N)$ in the additive sharing $\sum_{i=1}^n \varphi_i$. We denote $\varphi(N)$ by φ. In order to realize the $(t+1, n)$ threshold Paillier cryptosystem, the parties perform the following steps.

Key Generation Phase:

1. The n parties perform the distributed RSA modulus generation protocol in Sect. 4.3, and the additional protocol in Sect. 4.4.1 to make sure that Q_{N^2} is cyclic. As a result, they publish N and have φ in the additive sharing $\varphi = \sum_{i=1}^n \varphi_i$ over the integers.
2. Each party P_i picks up a random $\beta_i \in \mathbb{Z}_N^*$. By using the BGW protocol modulo N, the parties compute and reveal
$$\theta' = \beta\varphi = \left(\sum_{i=1}^n \beta_i\right)\left(\sum_{i=1}^n \varphi_i\right) \bmod N$$
as they computed $N = (\sum_{i=1}^n p_i)(\sum_{i=1}^n q_i) \bmod P$. θ' becomes part of PK.
3. Using the same β_i's, the parties compute the shares of
$$\Delta^4\beta\varphi = \left(\sum_{i=1}^n \Delta^2\beta_i\right)\left(\sum_{i=1}^n \Delta^2\varphi_i\right)$$

over the integers by the BGW protocol. The parties, however, do not reveal $\Delta^4\beta\varphi$ itself. Instead, they convert the $2t$-degree polynomial sharing of $\Delta^4\beta\varphi$ into the additive sharing of $\Delta^4\beta\varphi$ over the integers (poly-to-sum) where each share s_i satisfies (assuming $n = 2t + 1$)

$$\Delta^4\beta\varphi = \sum_{i=1}^{n} s_i.$$

Because $\beta_i < N$, $\sum_{i=1}^{n}\beta_i < nN$, $|\varphi_i| < N$, and $\sum_{i=1}^{n}\varphi_i < N$, each share of β_i and φ_i is bounded by $\Delta(\Delta N + t \times \Delta^2 NKn^t)(\approx t\Delta^3 NKn^t)$ where K is chosen such that $\frac{t}{K}$ is negligible. Therefore, each s_i is roughly bounded by $((t\Delta^3 NKn^t) \times n)^2$.

4. P_i generates a polynomial,

$$f_i(X) = \Delta(\Delta s_i + a_{i,1}X + \cdots + a_{i,t}X^t),$$

to reshare s_i in the $(t + 1, n)$ threshold secret sharing over the integers where $a_{i,j} \in_R [-K\Delta^2((t\Delta^3 NKn^t) \times n)^2, K\Delta^2((t\Delta^3 NKn^t) \times n)^2]$.

Note that if we say $f(X) = \sum_{i=1}^{n} f_i(X)$, then $f(0) = \Delta^6\beta\varphi$.

5. P_i computes $f_i(j)$ for $1 \le j \le n$, and sends $f_i(j)$ to P_j through the secure channel.
6. After receiving the shares from other parties, P_j computes $f(j) = \sum_{i=1}^{n} f_i(j)$. $f(j)$ is the secret share of $(t + 1, n)$ secret sharing of $\Delta^6\beta\varphi$.
7. The parties agree on k'-tuple verification keys $\langle VK_1 \bmod N^2, \ldots, VK_{k'} \bmod N^2\rangle$ chosen at random, and each party S_i publishes k'-tuple VK_i's

$$\langle VK_{1,i} = (VK_1)^{\Delta f(i)} \bmod N^2, \ldots, VK_{k',i} = (VK_{k'})^{\Delta f(i)} \bmod N^2\rangle.$$

Decryption Phase:

Now the parties decrypt a ciphertext $c = g^M x^N \bmod N^2$ where M is the plaintext.

1. P_i publishes the partial decryption $c_i = c^{2\Delta f(i)} \bmod N^2$ by using its secret share. Also P_i publishes the zero knowledge proofs for

$$f(i) = \log_{c^{4\Delta}}(c_i)^2 = \log_{(VK_1)^\Delta} VK_{1,i} = \cdots = \log_{(VK_{k'})^\Delta} VK_{k',i}.$$

We accept only the partial decryptions with the valid zero knowledge proofs. The reason

we use $f(i) = \log_{c^{4\Delta}}(c_i)^2$ instead of $f(i) = \log_{c^{2\Delta}} c_i$ is to make sure that we are working in Q_{N^2}.

2. By combining $t + 1$ valid partial decryptions of the subset S of the parties, we can obtain

$$M = L(\prod_{i \in S} c_i^{2\mu_i} \bmod N^2) \times \frac{1}{4\Delta^8 \theta'} \bmod N$$

where $\mu_i = \Delta \times \lambda_{0,i}^S \in \mathbb{Z}$, and $L(u) = \frac{u-1}{N}$.

Proof of correctness:

$$\begin{aligned}
\prod_{i \in S} c_i^{2\mu_i} &= c^{4\Delta \sum_{i \in S} f(i)\mu_i} \\
&= c^{4\Delta \sum_{i \in S} \Delta f(i)\lambda_{0,i}^S} \\
&= c^{4\Delta^8 \beta\varphi} \\
&= (g^M x^N)^{4\Delta^8 \beta\varphi} \\
&= g^{4\Delta^8 \beta\varphi M} \quad (\because {}^{\forall} x, \quad x^{N\varphi} = 1 \bmod N^2) \\
&= 1 + 4\Delta^8 \beta\varphi MN \bmod N^2 \quad (\because g = N + 1).
\end{aligned}$$

Therefore,

$$L(\prod_{i \in S} c_i^{2\mu_i} \bmod N^2) = 4\Delta^8 \beta\varphi M = M \times 4\Delta^8 \theta' \bmod N.$$

Δ and θ' are public information. Thus, we have

$$M = L(\prod_{i \in S} c_i^{2\mu_i} \bmod N^2) \times \frac{1}{4\Delta^8 \theta'} \bmod N.$$

□

Chapter 5

Concluding Remarks

Every story has a happy ending and if the ending is not happy,
then it means the story has not ended yet.
— Author Unknown

In this dissertation, we handled two main subjects among privacy applications, which are called Ciphertext-Policy Attribute-Based Encryption and Multiparty Computation respectively and we realized the CP-ABE schemes where ciphertext policies can be hidden and efficient MPC protocols. These techniques are useful and can be deployed in the real world to solve the privacy issues. However, there is still room for improvement and we briefly mention a number of possibilities for further research.

Ciphertext-Policy Attribute-Based Encryption. The first CP-ABE scheme [BSW07] had a security proof only in the generic group model. Therefore, it was important to seek for CP-ABE schemes that have security proofs based on standard number theoretic assumptions. Though the scheme in [CN07] is provably secure, it can support only a limited type of access structure for ciphertext policies and the recent line of work [GJP+08, Wat08] offers provably secure CP-ABE schemes with more expressive access structures. In Chapter 2, we have discussed the two CP-ABE schemes that have the additional property that ciphertext policies can be hidden and it may be interesting to seek for constructions based on [GJP+08, Wat08] in which we can hide ciphertext policies.

To support more expressive non-monotonic access structures and handle negated attributes efficiently, it will be useful to construct a provably secure CP-ABE scheme with non-monotonic and hidden access structures by using the techniques from [OSW07, SW08]. If it is possible, we will be able to reduce the ciphertext size.

The scheme in [KSW08] can be more expressive than our constructions, but the ciphertext size can be $O(d^n)$ where $d \geq 2$ and n is the number of attributes when it realizes the full expressibility. Thus it will be challenging to construct a provably secure CP-ABE scheme with advanced access structures and small ciphertext size.

Shi et al. [SBC+07] achieved a predicate encryption scheme supporting wider range queries with a relaxed security notion called match-revealing security, while the standard security notion used in this dissertation is called match-concealing security. Relaxing the security notion may be one of the promising approaches to constructing efficient encryption schemes with more advanced access structures.

Multiparty Computation. A protocol for integer division is an important primitive which is missing in this dissertation. In the division protocol, given shared secrets a and b, the parties compute a sharing of $\lfloor \frac{a}{b} \rfloor$. In [ACS02, FJ06], the division protocols were proposed to perform a modulo reduction. If the range of the shared secrets a and b is known in advance, then the comparison protocol can be executed multiple times to compute $\lfloor \frac{a}{b} \rfloor$ or a protocol for computing a sharing of $\lfloor \frac{2^k}{b} \rfloor$ can be used to obtain $\lfloor \frac{a}{b} \rfloor$ where modulo reductions are assumed to be repeated and k is appropriately chosen. If we can come up with a much more efficient protocol than these known techniques, it will be useful and may be applicable to generating shared safe primes.

We described our protocols in the honest-but-curious model. However, it will be important to make our protocols robust against active adversaries and it is a non-trivial task if we want to make them robust without losing too much efficiency. To make our protocols robust, we may need new different techniques as the robust version [OK05] of [ACS02].

We did not assume that shared secrets were within the known limited range, but by assuming so, we may be able to come up with more efficient protocols for more specific situations without adhering to constant-round protocols. The line of work [Tof05, BDJ+06, FJ06, Tof07, BCD+08,

DT08, VIFF] shows the feasibility of such MPC protocols and it will be challenging and meaningful to solve large-scale real-world problems with a working implementation of practical and special-purpose protocols as the aims of [VIFF, SSCM].

References

[ACD+06] M. Abdalla, D. Catalano, A. Dent, J. Malone-Lee, G. Neven, and N. Smart, "Identity-based encryption gone wild," Proc. International Colloquium on Automata, Languages and Programming (ICALP), LNCS 4052, pp.300–311, Springer-Verlag, 2006.

[ACS02] J. Algesheimer, J. Camenisch, and V. Shoup, "Efficient computation modulo a shared secret with application to the generation of shared safe-prime products," Proc. Crypto 2002, LNCS 2442, pp.417–432, Springer-Verlag, 2002.

[AC06] T. Atkinson and M.C. Silaghi, "An efficient way to access an array at a secret index," Cryptology ePrint Archive 2006/157, 2006.

[BB89] J. Bar-Ilan and D. Beaver, "Non-cryptographic fault-tolerant computing in a constant number of rounds of interaction," Proc. ACM Symposium on Principles of Distributed Computing (PODC 1989), pp.201–209, 1989.

[BFP+01] O. Baudron, P.-A. Fouque, D. Pointcheval, G. Poupard, and J. Stern, "Practical multi-candidate election system," Proc. 20th ACM Symposium on Principles of Distributed Computing (PODC 2001), pp.274–283, 2001.

[Bau02] E.P. Bautista, "A square root bound on the minimum distance of some quadratic double circulant codes over F_5 and F_7,"
Dissertation available from `http://mathsci.math.admu.edu.ph/~banjo/`

[BMR90] D. Beaver, S. Micali, and P. Rogaway, "The round complexity of secure protocols," Proc. 22nd ACM Symposium on Theory of Computing (STOC 1990), pp.503–513, 1990.

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorem for non-cryptographic fault-tolerant distributed computation," Proc. 20th Annual ACM Symposium on Theory of Computing (STOC 1988), pp.1–10, 1988.

[BSW07] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext-policy attribute-based encryption,” Proc. IEEE Symposium on Security and Privacy, pp.321–334, 2007.

[BCD+08] P. Bogetoft, D.L. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J.D. Nielsen, J.B. Nielsen, K. Nielsen, J. Pagter, M. Schwartzbach, and T. Toft, “Multiparty computation goes live,” Cryptology ePrint Archive 2008/068, 2008.

[BDJ+06] P. Bogetoft, I. Damgård, T. Jakobsen, K. Nielsen, J. Pagter, and T. Toft, “A practical implementation of secure auctions based on multiparty integer computation,” Financial Cryptography 2006, LNCS 4107, pp.142–147, Springer-Verlag, 2006.

[BB04] D. Boneh and X. Boyen, “Short signatures without random oracles,” Proc. Eurocrypt 2004, LNCS 3027, pp.56–73, Springer-Verlag, 2004.

[BBG05] D. Boneh, X. Boyen, and E.-J. Goh, “Hierarchical identity based encryption with constant size ciphertext,” Proc. Eurocrypt 2005, LNCS 3494, pp.440–456, Springer-Verlag, 2005.

[BBS04] D. Boneh, X. Boyen, and H. Shacham, “Short group signatures,” Proc. Crypto 2004, LNCS 3152, pp.41–55, Springer-Verlag, 2004.

[BF97] D. Boneh and M. Franklin, “Efficient generation of shared RSA keys,” Proc. Crypto 1997, LNCS 1233, pp.425–439, Springer-Verlag, 1997.

[BF01] D. Boneh and M. Franklin, “Identity-based encryption from the Weil pairing,” Proc. Crypto 2001, LNCS 2139, pp.213–229, Springer-Verlag, 2001.

[BW07] D. Boneh and B. Waters, “Conjunctive, subset, and range queries on encrypted data,” Proc. Theory of Cryptography Conference (TCC 2007), LNCS 4392, pp.535–554, Springer-Verlag, 2007.

[BW06] X. Boyen and B. Waters, “Anonymous hierarchical identity-based encryption (without random oracles),” Proc. Crypto 2006, LNCS 4117, pp.290–307, Springer-Verlag, 2006.

[Cac03] C. Cachin, “An asynchronous protocol for distributed computation of RSA inverses and its applications,” Proc. ACM Symposium on Principles of Distributed Computing (PODC 2003), pp.153–162, 2003.

[CHL05] J. Camenisch, S. Hohenberger, and A. Lysyanskaya, “Compact e-cash,” Proc. Eurocrypt 2005, LNCS 3494, pp.302–321, Springer-Verlag, 2005.

[CHK03] R. Canetti, S. Halevi, and J. Katz, “A forward-secure public-key encryption scheme,” Proc. Eurocrypt 2003, LNCS 2656, pp.255–271, Springer-Verlag, 2003.

[CHK04] R. Canetti, S. Halevi, and J. Katz, “Chosen-ciphertext security from identity-based encryption,” Proc. Eurocrypt 2004, LNCS 3027, pp.207–222, Springer-Verlag, 2004.

[CGS00] D. Catalano, R. Gennaro, and S. Halevi, “Computing inverses over a shared secret modulus,” Proc. Eurocrypt 2000, LNCS 1807, pp.190–207, Springer-Verlag, 2000.

[CFL83a] A.K. Chandra, S. Fortune, and R.J. Lipton, “Lower bounds for constant depth circuits for prefix problems,” Proc. International Colloquium on Automata, Languages and Programming (ICALP), LNCS 154, pp.109–117, Springer-Verlag, 1983.

[CFL83b] A.K. Chandra, S. Fortune, and R.J. Lipton, “Unbounded fan-in circuits and associative functions,” Proc. 15th ACM Symposium on Theory of Computing (STOC 1983), pp.52–60, 1983.

[Cha07] M. Chase, “Multi-authority attribute based encryption,” Proc. Theory of Cryptography Conference (TCC 2007), LNCS 4392, pp.515–534, Springer-Verlag, 2007.

[CCD88] D. Chaum, C. Crêpeau, and I. Damgård, “Multi-party unconditionally secure protocols,” Proc. ACM Symposium on Theory of Computing (1988), pp.11–19, 1988.

[CN07] L. Cheung and C. Newport, “Provably secure ciphertext policy ABE,” Proc. ACM Conference on Computer and Communications Security (CCS 2007), pp.456–465, 2007.

[CD01] R. Cramer and I. Damgård, “Secure distributed linear algebra in a constant number of rounds,” Proc. Crypto 2001, LNCS 2139, pp.119–136, Springer-Verlag, 2001.

[CDI05] R. Cramer, I. Damgård, and Y. Ishai, “Share conversion, pseudorandom secret sharing and applications to secure computation,” Proc. 2nd Theory of Cryptography Conference (TCC 2007), LNCS 3378, pp.342–362, Springer-Verlag, 2005.

[CDN01] R. Cramer, I. Damgård, and J.B. Nielsen, “Multiparty computation from threshold homomorphic encryption,” Proc. Eurocrypt 2001, LNCS 2045, pp.280–300, Springer-Verlag, 2001.

[DD05] I. Damgård and K. Dupont, “Efficient threshold RSA signatures with general moduli and no extra assumptions,” Proc. Theory and Practice of Public-Key Cryptography (PKC 2005), LNCS 3386, pp.346–361, Springer-Verlag, 2005.

[DFK+06] I. Damgård, M. Fitzi, E. Kiltz, J.B. Nielsen, and T. Toft, "Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation," Proc. 3rd Theory of Cryptography Conference (TCC 2007), LNCS 3876, pp.285–304, Springer-Verlag, 2006.

[DI05] I. Damgård and Y. Ishai, "Constant-round multiparty computation using a black-box pseudorandom generator," Proc. Crypto 2005, LNCS 3621, pp.378–394, Springer-Verlag, 2005.

[DJ01] I. Damgård and M. Jurik, "A generalisation, a simplification and some applications of Paillier's probabilistic public-key system," Proc. Theory and Practice of Public-Key Cryptography (PKC 2001), LNCS 1992, pp.119–136, Springer-Verlag, 2001.

[DJ03] I. Damgård and M. Jurik, "A length-flexible threshold cryptosystem with applications," Proc. 8th Australasian Conference, Information Security and Privacy (ACISP 2003), LNCS 2727, pp.350–364, Springer-Verlag, 2003.

[DK01] I. Damgård and M. Koprowski, "Practical threshold RSA signatures without a trusted dealer," Proc. Eurocrypt 2001, LNCS 2045, pp.152–165, Springer-Verlag, 2001.

[DN03] I. Damgård and J.B. Nielsen, "Universally composable efficient multiparty computation from threshold homomorphic encryption," Proc. Crypto 2003, LNCS 2729, pp.247–264, Springer-Verlag, 2003.

[DT08] I. Damgård and R. Thorbek, "Efficient conversion of secret-shared values between different fields," Cryptology ePrint Archive 2008/221, 2008.

[ElG85] T. ElGamal, "A public key cryptosystems and a signature scheme based on discrete logarithm," Proc. Crypto 1984, LNCS 197, pp.10–18, Springer-Verlag, 1985.

[FKN94] U. Feige, J. Kilian, and M. Naor, "A minimal model for secure computation," Proc. 26th ACM Symposium on Theory of Computing (STOC 1994), pp.554–563, 1994.

[FPS00] P.-A. Fouque, G. Poupard, and J. Stern, "Sharing decryption in the context of voting or lotteries," Proc. Financial Cryptography 2000, LNCS 1962, pp.90–104, Springer-Verlag, 2000.

[FS01] P. A. Fouque, and J. Stern, "Fully distributed threshold RSA under standard assumptions," Proc. Asiacrypt 2001, LNCS 2248, pp.310–330, Springer-Verlag, 2001.

[FGMY97] Y. Frankel, P. Gemmell, P. Mackenzie, and M. Yung, “Optimal resilience proactive public-key Cryptosystems,” Proc. 38th IEEE Symposium on Foundation of Computer Science (FOCS), pp.384–393, 1997.

[FMY98] Y. Frankel, P. D. MacKenzie, and M. Yung, “Robust efficient distributed RSA-key generation,” Proc. 30th ACM Symposium on Theory of Computing (STOC 1998), pp.663–672, 1998.

[FGM07] M. Franklin, M. Gondree, and P. Mohassel, “Multi-party indirect indexing and applications,” Proc. Asiacrypt 2007, LNCS 4833, pp.283–297, Springer-Verlag, 2007.

[FGM07b] M. Franklin, M. Gondree, and P. Mohassel, “Improved efficiency for private stable matching,” Proc. Cryptographers’ Track at the RSA Conference (CT-RSA 2007), LNCS 4377, pp.163–177, Springer-Verlag, 2007.

[FJ06] S. L. From and T. Jakobsen, “Secure multi-party computation on integers,” Master’s Thesis, http://www.daimi.au.dk/~mas/uni/thesis/index.html, 2006

[GJKR96] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, “Robust and efficient sharing of RSA functions,” Proc. Crypto 1996, LNCS 1109, pp.157–172, Springer-Verlag, 1996.

[GRR98] R. Gennaro, M.O. Rabin, and T. Rabin, “Simplified VSS and fast-track multiparty computations with applications to threshold cryptography,” Proc. 17th ACM Symposium on Principles of Distributed Computing (PODC 1998), pp.101–110, 1998.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson, “How to play any mental game or a complete theorem for protocols with honest majority,” Proc. 19th ACM Symposium on Theory of Computing (STOC 1987), pp.218–229, 1987.

[Gol06] P. Golle, “A private stable matching algorithm,” Proc. Financial Cryptography and Data Security, LNCS 4107, pp.65–80, Springer-Verlag, 2006.

[GJP+08] V. Goyal, A. Jain, O. Pandey, and A. Sahai, “Bounded ciphertext policy attribute based encryption,” Proc. International Colloquium on Automata, Languages and Programming (ICALP), LNCS 5126, pp.579–591, Springer-Verlag, 2008.

[GPS+06] V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-based encryption for fine-grained access control of encrypted data,” Proc. ACM Conference on Computer and Communications Security (CCS 2006), pp.89–98, 2006.

[HN05] M. Hirt and J.B. Nielsen, “Upper bounds on the communication complexity of optimally resilient cryptographic multiparty computation,” Proc. Asiacrypt 2005, LNCS 3788, pp.79–99, 2005.

[HN06] M. Hirt and J.B. Nielsen, “Robust multiparty computation with linear communication complexity,” Proc. Crypto 2006, LNCS 4117, pp.463–482, 2006.

[JJ00] M. Jakobsson and A. Juels, “Mix and match: secure function evaluation via ciphertexts,” Proc. Asiacrypt 2000, LNCS 1976, pp.162–177, 2000.

[JA03] H. Jordan and G. Alaghband “Fundamentals of parallel processing,” Prentice Hall, 2003.

[Jou00] A. Joux, “A one round protocol for tripartite Diffie-Hellman,” Proc. ANTS-IV, pp.385–394, 2000.

[KTS07] A. Kapadia, P. P. Tsang, and S. W. Smith, “Attribute-based publishing with hidden credentials and hidden policies,” Proc. Network & Distributed System Security Symposium (NDSS 2007), pp.179–192, 2007.

[KSW08] J. Katz, A. Sahai, and B. Waters, “Predicate encryption supporting disjunctions, polynomial equations, and inner products,” Proc. Eurocrypt 2008, LNCS 4965, pp.146–162, Springer-Verlag, 2008.

[LF80] R. Ladner and M. Fischer, “Parallel prefix computation,” Journal of the Association for Computing Machinery vol.27, pp.831–838, 1980.

[LS08] D. Lubicz and T. Sirvent, “Attribute-based broadcast encryption scheme made efficient,” Proc. Africacrypt 2008, LNCS 5023, pp.325–342, Springer-Verlag, 2008.

[MWB99] M. Malkin, T. Wu, and D. Boneh, “Experimenting with shared RSA key generation,” Proc. Internet Society’s 1999 Symposium on Network and Distributed System Security (SNDSS 1999), pp.43–56, 1999.

[MNT01] A. Miyaji, M. Nakabayashi, and S. Takano, “New explicit conditions of elliptic curve traces for FR-reductions,” IEICE Trans., Fundamentals, E84-A(5), pp.1234–1243, 2001.

[MF06] P. Mohassel and M.K. Franklin, “Efficient polynomial operations in the shared-coefficients setting,” Proc. Theory and Practice of Public-Key Cryptography (PKC 2006), LNCS 3958, pp.44–57, Springer-Verlag, 2006.

[NN01] M. Naor and K. Nissim, “Communication preserving protocols for secure function evalu-

ation," Proc. 33rd ACM Symposium on Theory of Computing (STOC 2001), pp.590–599, 2001.

[Nec94] V. I. Nechaev, "On the complexity of a deterministic algorithm for a discrete logarithm," Mat. Zametki, 55(2), pp.91–101, 189, 1994.

[NO07] Takashi Nishide, Kazuo Ohta, "Multiparty computation for interval, equality, and comparison without bit-decomposition protocol," *10th International Conference on Theory and Practice of Public-Key Cryptography (PKC 2007)*, Beijing, China, April 16–20, 2007. Lecture Notes in Computer Science (LNCS) 2250, pp.343–360, Springer-Verlag, 2007.

[NYO08] Takashi Nishide, Kazuki Yoneyama, Kazuo Ohta, "Attribute-based encryption with partially hidden encryptor-specified access structures," *6th International Conference on Applied Cryptography and Network Security (ACNS 2008)*, New York, USA, June 3–6, 2008. Lecture Notes in Computer Science (LNCS) 5037, pp.111–129, Springer-Verlag, 2008.

[NZ05] K. Nissim and R. Zivan, "Secure DisCSP protocols - from centralized towards distributed solutions," Proc. 6th Workshop on Distributed Constraint Reasoning (DCR-05), 2005.

[OK05] E. Ong and J. Kubiatowicz, "Optimizing robustness while generating shared secret safe primes," Proc. Theory and Practice of Public-Key Cryptography (PKC 2005), LNCS 3386, pp.120–137, Springer-Verlag, 2005.

[OSW07] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," Proc. ACM Conference on Computer and Communications Security (CCS), pp.195–203, 2007.

[Pai99] P. Paillier, "Public-key cryptosystems based on composite degree residue classes," Proc. Eurocrypt 1999, LNCS 1666, pp.223–238, Springer-Verlag, 1999.

[Ped91] T. Pedersen, "A threshold cryptosystem without a trusted party," Proc. Eurocrypt 1991, LNCS 547, pp.522–526, Springer-Verlag, 1991.

[Per52] O. Perron, "Bemerkungen über die verteilung der quadratischen reste," Mathematische Zeitschrift, vol.56, no.2, pp.122–130, 1952.

[Rab98] T. Rabin, "A simplified approach to threshold and proactive RSA," Proc. Crypto 1998,

LNCS 1462, pp.89–104, Springer-Verlag, 1998.

[RSA78] R. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signature and public-key cryptosystems,” Communication of the ACM, vol.21, no.2, pp.120–126, 1978.

[SW05] A. Sahai and B. Waters, “Fuzzy identity-based encryption,” Proc. Eurocrypt 2005, LNCS 3494, pp.457–473, Springer-Verlag, 2005.

[SW08] A. Sahai and B. Waters, “Revocation systems with very small private keys,” Cryptology ePrint Archive 2008/309, 2008.

[ST06] B. Schoenmakers and P. Tuyls, “Efficient binary conversion for Paillier encrypted values,” Proc. Eurocrypt 2006, LNCS 4004, pp.522–537, Springer-Verlag, 2006.

[Sch80] J.T. Schwartz, “Fast probabilistic algorithms for verification of polynomial identities,” J. ACM, vol.27, no.4, pp.701–717, 1980.

[Sco02] M. Scott, “Authenticated ID-based key exchange and remote log-in with simple token and PIN number,” Cryptology ePrint Archive 2002/164, 2002.

[Sha79] A. Shamir, “How to share a secret,” Communications of ACM, vol.22, no.11, pp.612–613, 1979.

[SBC+07] E. Shi, J. Bethencourt, H. Chan, D. Song, and A. Perrig, “Multi-dimensional range query over encrypted data,” Proc. IEEE Symposium on Security and Privacy, pp.350–364, 2007.

[Sho97] V. Shoup, “Lower bounds for discrete logarithms and related problems,” Proc. Eurocrypt 1997, LNCS 1233, pp.256–266, Springer-Verlag, 1997.

[Sho00] V. Shoup, “Practical threshold signatures,” Proc. Eurocrypt 2000, LNCS 1807, pp.207–220, Springer-Verlag, 2000.

[SSCM] SecureSCM Project. http://securescm.org/

[Tof05] T. Toft, “Secure integer computation with applications in economy,” Technical Report available from http://www.dinapigen.dk/TOMAS/publications.html

[Tof07] T. Toft, “Primitives and applications for multi-party computation,” Dissertation available from http://www.dinapigen.dk/TOMAS/publications.html

[VIFF] The Virtual Ideal Functionality Framework. http://viff.dk/

[Wat08] B. Waters, “Ciphertext-policy attribute-based encryption: an expressive, efficient, and

provably secure realization," Cryptology ePrint Archive 2008/290, 2008.

[Yao82] A.C. Yao, "Protocols for secure computation," Proc. 23rd IEEE Symposium on Foundation of Computer Science (FOCS), pp.160–164, 1982.

[Yao86] A.C. Yao, "How to generate and exchange secrets," Proc. 27th IEEE Symposium on Foundation of Computer Science (FOCS), pp.162–167, 1986.

[Zip79] R. Zippel, "Probabilistic algorithms for sparse polynomials," Proc. EUROSAM 1979, LNCS 72, pp.216–226, Springer-Verlag, 1979.

Acknowledgements

First of all I would like to thank my advisor, Kazuo Ohta for his endless encouragement, optimism, enthusiasm, advice, and patience. He was always available as a reliable oracle and gave me many hints on doing research whenever I got stuck on research problems and things were so unclear. He always understood my situation and gave me timely advice that helped me pick good directions even when we seemed to be in adversity. His casual word “Let’s join the race” led me to come up with the first result, the comparison protocol. I am so indebted to him for his guidance during my Ph.D. study.

I am also grateful to Noboru Kunihiro for teaching me the basics of IBE that was very useful for my research on ABE and sharing a room with me at PKC 2007 that I enjoyed so much.

My special thanks go to Kazuki Yoneyama for sharing with me his broad knowledge on cryptography, not ignoring my stupid questions, and collaboration.

I also thank Tomas Toft for the discussion at TCC 2006 and giving me his idea of the LSB protocol included in this dissertation.

I express my gratitude to Ming-Deh Huang for giving me an opportunity to do research on cryptography under his supervision during my master’s study, which still preoccupies me and led me to this dissertation.

I also extend my gratitude to the members of my dissertation committee for their willingness to serve on the committee and helpful comments on this dissertation.

Being a member of this laboratory allowed me to have chances to interact with many good people in the field of information security that I would not have met if I had not taken a step into research on cryptography, who include Tetsuya Izu, Mitsugu Iwamoto, Bagus Santoso, Yoshikazu Hanatani, Yasuhiko Hiehata, Seiji Okuaki, Eiichiro Fujisaki, and Miyako Ohkubo. Especially I

thank Seiji Okuaki for sharing with me the similar struggles as a part-time Ph.D. student, Eiichiro Fujisaki for having time for me and his advice, and Miyako Ohkubo for giving me a chance to give a talk in front of Prof. Adi Shamir at IPA, which was a memorable moment.

I thank Fumiko Sekiguchi for fun chats and helping me cope with non-cryptographic paperwork at the university.

I thank the product development team that I am a member of in the company for allowing me to take days off to attend the academic conferences when we were so busy.

Finally I thank my family for always being supportive of my life.

Takashi Nishide,

Shinmaruko, September 13, 2008.

List of Publications Related to the Dissertation

Related Publications

Journal Papers

1. Takashi Nishide, Kazuo Ohta, “Constant-Round Multiparty Computation for Interval Test, Equality Test, and Comparison”, IEICE Transactions on Fundamentals, Vol. E90–A, No.5, pp.960–968, May 2007.
2. Takashi Nishide, Kazuki Yoneyama, Kazuo Ohta, “Attribute-Based Encryption with Partially Hidden Ciphertext Policies”, IEICE Transactions on Fundamentals (To Appear), January 2009.

Refereed Conference Papers (with Formal Proceedings)

1. Takashi Nishide, Kazuo Ohta, “Multiparty Computation for Interval, Equality, and Comparison without Bit-Decomposition Protocol”, *10th International Conference on Theory and Practice of Public-Key Cryptography (PKC 2007)*, Beijing, China, April 16–20, 2007. Lecture Notes in Computer Science (LNCS) 2250, pp.343–360, Springer-Verlag, 2007.
2. Takashi Nishide, Kazuki Yoneyama, Kazuo Ohta, “Attribute-Based Encryption with Partially Hidden Encryptor-Specified Access Structures”, *6th International Conference on Applied Cryptography and Network Security (ACNS 2008)*, New York, USA, June 3–6, 2008.

Lecture Notes in Computer Science (LNCS) 5037, pp.111–129, Springer-Verlag, 2008.

Patents

1. 西出隆志, 太田和夫, “秘密分散情報処理システム”, 特開 2008–20871(P2008–20871A)

List of All Publications

Journal Papers

1. Takashi Nishide, Kazuo Ohta, “Constant-Round Multiparty Computation for Interval Test, Equality Test, and Comparison”, IEICE Transactions on Fundamentals, Vol. E90–A, No.5, pp.960–968, May 2007.
2. Takashi Nishide, Kazuki Yoneyama, Kazuo Ohta, “Attribute-Based Encryption with Partially Hidden Ciphertext Policies”, IEICE Transactions on Fundamentals (To Appear), January 2009.

Refereed Conference Papers (with Formal Proceedings)

1. Takashi Nishide, Kazuo Ohta, “Multiparty Computation for Interval, Equality, and Comparison without Bit-Decomposition Protocol”, *10th International Conference on Theory and Practice of Public-Key Cryptography (PKC 2007)*, Beijing, China, April 16–20, 2007. Lecture Notes in Computer Science (LNCS) 2250, pp.343–360, Springer-Verlag, 2007.
2. Takashi Nishide, Kazuki Yoneyama, Kazuo Ohta, “Attribute-Based Encryption with Partially Hidden Encryptor-Specified Access Structures”, *6th International Conference on Applied Cryptography and Network Security (ACNS 2008)*, New York, USA, June 3–6, 2008. Lecture Notes in Computer Science (LNCS) 5037, pp.111–129, Springer-Verlag, 2008.

Patents

1. 西出隆志, 太田和夫, “秘密分散情報処理システム”, 特開 2008–20871(P2008–20871A)

Non-refereed Papers

1. Takashi Nishide, Kazuo Ohta, “How to Compare Two Polynomially Shared Secrets Privately”, Symposium on Cryptography and Information Security (SCIS 2006), January 2006.
2. Takashi Nishide, Kazuki Yoneyama, Kazuo Ohta, “Attribute-Based Encryption with Partially Hidden Ciphertext Policies”, IEICE Technical Report, ISEC2007-125, pp.93–100, December 2007.
3. Takashi Nishide, Kazuki Yoneyama, Kazuo Ohta, “Attribute-Based Encryption with Hidden Encryptor-Specified Policies”, Symposium on Cryptography and Information Security (SCIS 2008), January 2008.

Author Biography

Takashi Nishide was born in Osaka, Japan, on June 5, 1973. He received his B.S. degree in information science from the University of Tokyo, Tokyo, Japan, in March 1997, and his M.S. degree in computer science from the University of Southern California, California, USA, in 2003, respectively. Since 1997, he has been working at Hitachi Software Engineering Co., Ltd. and engaged in developing security products. His reserch interests include public-key cryptosystems and cryptographic protocols. Since April 2005, he had been a Ph.D. student at the Graduate School of Electro-Communications, the University of Electro-Communications, working towards his Ph.D. degree as a part-time student and finished the doctor course in March 2008. Mr. Nishide is a member of the International Association for Cryptology Research (IACR).