OBLIVIOUS HANDSHAKES AND SHARING OF SECRETS OF PRIVACY-PRESERVING MATCHING AND AUTHENTICATION PROTOCOLS

A Dissertation

by

PU DUAN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2011

Major Subject: Computer Science

OBLIVIOUS HANDSHAKES AND SHARING OF SECRETS OF PRIVACY-PRESERVING MATCHING AND AUTHENTICATION PROTOCOLS

A Dissertation

by

PU DUAN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Jyh-Charn Liu
Committee Members,	Guofei Gu
	Anxiao Jiang
	Bojan Popov
Head of Department,	Valerie E. Taylor

May 2011

Major Subject: Computer Science

ABSTRACT

Oblivious Handshakes and Sharing of Secrets of Privacy-Preserving Matching and Authentication Protocols. (May 2011)

Pu Duan, B.S., Xi’an Jiaotong University, China

Chair of Advisory Committee: Dr. Jyh-Charn Liu

The objective of this research is focused on two of the most important privacy-preserving techniques: privacy-preserving element matching protocols and privacy-preserving credential authentication protocols, where an *element* represents the information generated by users themselves and a *credential* represents a *group membership* assigned from an independent *central authority* (CA). The former is also known as *private set intersection* (PSI) protocol and the latter is also known as *secret handshake* (SH) protocol. In this dissertation, I present a general framework for design of efficient and secure PSI and SH protocols based on similar message exchange and computing procedures to confirm “commonality” of their exchanged information, while protecting the information from each other when the commonalty test fails. I propose to use the *homomorphic randomization function* (HRF) to meet the privacy-preserving requirements, i.e., a common element/credential can be computed efficiently based on homomorphism of the function, and an uncommon element/credential is difficult to derive because of the randomization of the same function.

Based on the general framework, two new PSI protocols with linear computing and communication cost are proposed. The first protocol uses full homomorphic randomization function as the cryptographic basis, and the second one uses partial homomorphic randomization function. Both of them achieve element confidentiality and private set intersection. A new SH protocol is also designed based on the framework, which achieves unlinkability with a reusable pair of credential and pseudonym and the fewest number of bilinear mapping operations. I also propose to interlock the proposed PSI protocols and SH protocol to design new protocols with new security properties. When a PSI protocol is executed first and the matched elements are associated with the credentials in a following SH protocol, authenticity is guaranteed on matched elements. When a SH protocol is executed first and the verified credentials is used in a following PSI protocol, detection resistance and impersonation attack resistance are guaranteed on matching elements.

The proposed PSI and SH protocols are implemented to provide privacy-preserving inquiry matching service (PPIM) for social networking applications and privacy-preserving correlation service (PAC) of network security alerts. PPIM allows online social consumers to find partners with matched inquiries and verified group memberships, without exposing any information to unmatched parties. PAC allows independent network alert sources to find the common alerts without unveiling their local network information to each other.

DEDICATION

To *my wife and parents*

ACKNOWLEDGEMENTS

First, I would like to sincerely thank my advisor, Dr. Jyh-Charn Liu, for his support, guidance, trust and encouragement. He means more than an advisor to me. He helped me become mature, both personally and professionally. He encouraged me to follow my true heart and pursue the academic goal I always desired. I would like to thank Dr. Guofei Gu, Dr. Bojan Popov, and Dr. Anxiao Jiang for spending their valuable time and effort in serving on my committee. They gave me very helpful suggestions on improving the quality of my research.

I would also like to thank my labmates in the Real-time Distributed Systems Lab at TAMU. I am greatly in debt to Huajun Ying, Cheng Chung Tan, Weiqin Ma, Sanmin Liu, Peng Tao, Pu Shi and Hao Wang, for their help in various projects. I would like to thank Tak Cheung Lam, Hong Lu, Ming Zhang and Hai Xu for their generosity in sharing many ideas with me.

Last, I appreciate my beloved wife, my parents and my grandfather. Without their love, patience, and support, I would not have finished my Ph.D. study.

NOMENCLATURE

ECC	Elliptic Curve Cryptograpy
HRF	Homomorphic Randomization Function
PSI	Private Set Intersection
PAC	Privacy-Preserving Alert Correlation
PPIM	Privacy-Preserving Inquiry Matching
SH	Secret Handshake

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

Privacy concerns have become one of the most important security issues in many Internet applications, e.g., privacy leaks in social networking services [1][2][3][4][5], malicious vulnerability exploitation through public networking alerts sharing services [6][7][8][9], and identity attacks on a genomic database [10][11], etc. The dilemma of enjoying more convenient networking services vs. risking the compromise of more private information raises a key problem for researchers: How to provide practical and secure Internet services while protecting consumer privacy? To solve this problem, numerous *privacy-preserving applications* have been proposed in the literature [6][10][12][13][14][15][16][17][18][19][20]. Despite their different degrees of success, it is well recognized that most existing solutions face challenges of poor system scalability, high computing costs, and inadequate security protections [21][22][23].

Most of the above applications rely heavily on various cryptographic techniques to fulfill the designated functionality while preserving user privacy. Two of the most important techniques are privacy-preserving element matching protocols and privacy-preserving credential authentication protocols. They are designed to provide protection for user's two main kind of information: *element* as information that are generated by users themselves and *credential* as information that are assigned from dependent *central authority* (CA). The former, also known as *private set intersection* (PSI) protocol [24], aims to compute the intersection of elements provided by two parties without exposing

This dissertation follows the style of *IEEE Transactions on Networking.*

other information. Elements not in the intersection are kept secret. The latter, also known as *secret handshake* (SH) protocol [25], aims to verify/authenticate credentials of two parties. In this dissertation a credential represents the membership of a *group*. A SH protocol requires that two parties can confirm matching of affiliations if they have the same credentials (group memberships); otherwise, the affiliation of one party cannot be derived from the other party. These two subjects are widely studied, because improving associated computing and communication costs will have profound implications on a broad range of privacy-preserving applications like privacy-preserving genomic computation [10][18], privacy-preserving network alert correlation [6][12][14], and private email service [26].

In this dissertation, a general framework was proposed for design of efficient and secure privacy-preserving element matching and credential authentication protocols. Based on a thorough analysis of a broad range of algorithms, we concluded that they follow similar message exchange and computing procedures, and use similar techniques to confirm "commonality" of their exchanged information, while protecting the information from each other when the commonalty test fails. A *homomorphic randomization function* (HRF) was proposed to meet the conflicting requirement, i.e., common element/credential can be computed efficiently based on homomorphism of the function and uncommon element/credential are difficult to derive based on the randomization of the same function. A common ground of the two major fields was also introduced to streamline their system models and cryptographic bases, so that we can design more efficient algorithms for both types of applications. The holistic modeling of different privacy preserving protocols on the same crypto bases greatly simplifies the design of new and more efficient privacy-preserving protocols. That said, either a PSI or a SH protocol can be designed in three common phases: *Protocol Initialization* phase, *Secret Mapping* phase and *Result Computing* Phase. For a PSI or SH protocol executed between Alice and Bob, in the first phase Alice's secrets are processed by using the HRF to meet the security requirement. Then in the second phase Bob's secrets are combined with the Alice's randomized secrets by using the HRF, too. At last in the third phase the combined secrets are computed by Alice to find out the matched ones.

Based on the proposed general framework and homomorphic randomization function two new private set intersection protocols with linear computational and communication complexity were proposed for element matching. In our design Alice (Bob) directly takes elements as secret numbers and associates them with chosen large random numbers such that elements cannot be derived. Since the random numbers are generated through a homomorphic function, Alice (Bob) can modify Bob's (Alice's) associated values for privacy-preserving matching. While the first PSI protocol is based on full homomorphic randomization function, the second protocol is based on partial homomorphic randomization function, which is more secure against active attacks like falsification. The proposed solutions improve existing PSI protocols by reducing time complexity from $O(n^2)$ (n is the number of elements) [24] or $O(nL)$ (L is the average bit length of an element) [27] to $O(n)$ in honest-but-curious adversarial model.

Based on the framework and homomorphic randomization function a new unlinkable secret handshake protocol with reusable credential was proposed for credential authentication. In our design user's assigned pseudonyms are randomized by the homomorphic randomization function to provide reusability of credentials. The random number minimizes the correlation among authentication messages even they are produced by reuse of the same credential. The proposed solution improves existing unlinkable SH protocols by reducing communication complexity from infinite one-time-use credentials [25] to constant number of credentials and time complexity from 6 *bilinear mapping* operations [28] to 2 bilinear mapping operations.

The proposed protocols were also interlocked to provide authenticity on matching elements. In our first design a private set intersection protocol is executed first and matched elements are computed. Then a secret handshake protocol is executed for privacy-preserving authentication of both credentials and the matching elements. Our design associates the matching elements with group secret through the HRF and provides authenticity on the matching elements. In our second design a secret handshake protocol is executed first and assigned credentials are verified. Then a private set intersection is executed for privacy-preserving matching of both elements and the verified credentials. Our design utilizes the pairwise session key generated after a successful secret handshake and guarantees detection and impersonation attack resistance on the matching elements.

The proposed protocols were implemented to provide privacy-preserving inquiry matching service for social networking applications (e.g., Facebook [29]) and privacy-preserving correlation service of network security alerts. In the privacy-preserving social application we designed a privacy preserving inquiry matching (PPIM) system based on an end-user privacy control model. Online social service consumers first use non-privileged information to discover PPIM candidates on existing social networking services (SNS). The discovery results are then passed to a PPIM relay server by SNS, so that interested parties can engaged in PPIM transactions through exchange of encrypted inquiries between pairs. Using the proposed PSI protocol for matching of (encrypted) inquires, and the unlinkable secret handshaking protocol for membership verification, only the owner and her peer with a matched inquiry can know the real transaction content, but not any other users, including SNS providers. Experimental results show that the proposed scheme has scalable, reasonable computing costs, roughly 0.12-0.13 per request consisting of 10 records for the encryption strength equivalent to 1024-bit RSA. In the privacy-preserving alert correlation application we designed a privacy-preserving alert correlation (PAC) system to correlate network security alerts between semi-honest users, i.e., users who will correlate authentic alerts generated from their networks, but they may want to learn about others' alerts using passive attacks. Two different protocols based on the homomorphism of the elliptic curve cryptosystem (ECC) were implemented for PAC. The first protocol is the proposed partial homomorphism based PSI protocol, which is implemented for privacy-preserving matching of attributes of known formats in PAC, e.g., IP addresses, port numbers, etc. Expanding from the matching protocol, the second protocol is designed for privacy-preserving computation of the longest common sub-string (LCS) between two input strings. A major challenge is solved in the second protocol to protect locations of unmatched string elements while correctly identify the LCS. The LCS can be used for generating common contents of captured malware/intrusion payloads. We implemented a set of working prototypes of PAC to demonstrate privacy-preserving detection of attack patterns (e.g., attack chain) and attack contents (e.g., common worm signature) using alerts generated from Snort [30], Autograph [31], and a trace collected from a real-world honeynet. The experiments show that crypto-based correlation of large number of highly privacy sensitive alert data can be done at reasonable costs, e.g., 2 minutes for matching of two sets of 10k records.

II. A GENERAL FRAMEWORK OF PSI AND SH PROTOCOLS

In this section we propose a general framework to represent the interrelationship between privacy-preserving element matching protocols and privacy-preserving credential authentication protocols, where an element refers to a string generated by a user, and a credential refers to a ciphertext generated by an authority and issued to a user for subsequent off-line, mutual authentication of the membership of a group. The unified framework is used to illustrate the common structure for both the matching protocols and authentication protocols. We also analyze their relationship and illustrate how to utilize different mathematical functions to design different matching and authentication protocols under the same framework.

A. INTRODUCTION TO PRIVACY-PRESERVING PROTOCOLS

Private set interaction (PSI) protocols and secret handshaking (SH) protocols are two major families of privacy preserving matching protocols. In both protocols Bob and Alice exchange encoded messages to jointly compute certain type of commonality from the messages. For PSI, Bob and Alice can find out the intersection of their two sets of elements, without unveiling to each other the rest of unmatched elements. For SH, Bob and Alice can find out if they both belong to the same group using credentials that they received from central authorities (CA). Because of their different requirements in management of secrets (certain types of private data), these two types of protocols are usually studied independently. Direct management issues for secrets include generation, encryption-decryption, disclosure, and reuse of secrets. Moreover, indirect management issues, especially inference of using certain secrets for different transactions, i.e., aka linkability of transactions, are also considered important for certain applications.

In PSI a principal has total freedom in generation of new sets of elements for matching, yet in SH a principal must submit his/her identity to the CA, so that the CA can generate random numbers as his/her pseudonyms, and associates the pseudonyms with a value that represents the secret of the requested group to produce her credential. Here the users can only use her credential to perform authentication function and cannot derive the group secret from the assigned credential. Since both credentials and group memberships represent the same kind of information used to prove a user's affiliation, in the dissertation we will not differentiate between them. The objective of element matching between two parties is to find out the intersection of their input sets of elements (i.e., matched elements) without allowing one party to know the other party's unmatched elements. The objective of credential authentication between two parties is to verify whether or not their credentials have identical group secret, which is only known owned by a central authority. The privacy preserving property requires that the affiliation of one party cannot be derived from the other party, if they do not have the same group secret; otherwise, if they do, they can confirm matching of affiliations. In addition, it is not feasible to tell whether two executions of the handshake protocol were performed by the same party or not, even if both of them were successful.

Based on a high level analysis of a broad range of algorithms [24][25][26][27][28][32][33][34][35][36][37][38][39], one can conclude that they follow similar message exchange and computing procedures. They also use similar techniques to confirm “commonality” of their exchanged information, while protecting the information from each other when the commonalty test fails, as their common security requirements. The objective of this research is to explore the common ground of the two major fields to streamline their system models and cryptographic bases, so that we can design more efficient algorithms for both types of applications. More specifically, we propose to use homomorphic randomization function to meet the common conflicting requirement, i.e., common element/credential can be computed efficiently based on homomorphism of the function and uncommon element/credential are difficult to derive based on the randomization of the same function. As it will become clear later, holistic modeling of different privacy preserving protocols on the same crypto bases simplifies the design of new and more efficient privacy-preserving protocols. That said, either a PSI or a SH protocol can be designed in three common phases: *Protocol Initialization* phase, *Secret Mapping* phase and *Result Computing* Phase.

B. NOTATIONS OF PRIVACY-PRESERVING PROTOCOLS

Before going into the details of privacy-preserving protocols, we introduce the notation in Tables 1, 2, 3 and 4. In this research we assume that two parties Alice and Bob interact with each other to execute privacy-preserving element matching protocols and credential authentication protocols. In the follows we first present a definition list of all terms required in our privacy-preserving protocols.

Table 1 Definitions of general terms.

Group	Any number of parties considered as a unit with a common secret
Group Secret	A large number that represents the membership of a group
Identity	A party’s real name that is used to acquire credentials and pseudonyms
Pseudonym	An arbitrary string that represents the identity of a group member from CA
Credential	A large number that consists of a group secret and a pseudonym, which is used to prove genuineness of a group membership associated with the specific pseudonym
Central Authority	The authority who assigns credentials and pseudonyms to parties
Element	An arbitrary string generated by a party that belongd to a given set
Element Set	A collection of elements classed together
Intersection	The set of elements that two element sets have in common

Table 2 Notations of general parameters.

Central Authority (CA)	An authority who assigns credentials and pseudonyms to parties
Alice and Bob	Two legitimate parties who execute the protocols
Eve	Passive adversary
Mallory, Mallet, Trudy	Active adversaries
i, j, k, n, m, w	Six positive integers that represent numbers of elements, group secrets, pseudonyms, etc
τ	Security parameter
q	A large prime number
G_1	An additive cyclic group of prime order q
P_B	Base element (generator) of G_1
G_2	A multiplicative cyclic group of prime order q
H_1	A collision-free hash function that maps a string to an element in G_1
H_2	A collision-free hash function that maps a string with arbitrary length to a string with fixed length
e	A bilinear map
s_A, s_B	Alice's and Bob's secrets, respectively
r_{A1}, r_{A2}, r_{A3}	Alice's chosen random numbers
r_{B1}, r_{B2}, r_{A3}	Bob's chosen random numbers

Table 3 Notations of privacy-preserving credential authentication protocols.

$G = (g_1, \ldots, g_w)$	CA's set of group secrets
g_i, g_j	i-th and j-th secret in G
g_A, g_B	the group secret of Alice's and Bob's single group membership
$ID_A = (id_{A1}, \ldots, id_{An})$, $ID_B = (id_{B1}, \ldots, id_{Bm})$	Alice's and Bob's assigned pseudonym set, respectively
id_{Ai}, id_{Bj}	i-th and j-th assigned pseudonyms of ID_A and ID_B, respectively
id_A, id_B	Alice's and Bob's single pseudonym
$CRED_A = (cred_{A1}, \ldots, cred_{An})$, $CRED_B= (cred_{B1}, \ldots, cred_{Bm})$,	Alice's and Bob's assigned credential sets, respectively
$cred_{Ai}$, $cred_{Bj}$	i-th and j-th credentials of $CRED_A$ and $CRED_B$, respectively
$cred_A$, $cred_B$	Alice's and Bob's single credential , respectively
$cred_{Ai} = f(id_{Ai}, g_A)$, $cred_{Bj} = f(id_{Bj}, g_B)$	f presents the function that associates a group secret with a pseudonym
$rID_A = (rd_{A1}, \ldots, rd_{An})$, $rID_B = (rd_{B1}, \ldots, rd_{Bm})$	Alice's and Bob's homomorphicly randomized pseudonym sets, respectively
rd_{Ai}, rd_{Bj}	i-th and j-th homomorphicly randomized pseudonyms in rID_A and rID_B
rd_A, rd_B	Alice's and Bob's single homomorphicly randomized pseudonym
v_A, v_B	Alice's and Bob's credential verification values, respectively
$v_{A(B)}$, $v_{B(A)}$	Alice's verification value computed by Bob and Bob's verification value computed by Alice, respectively
V_{AB}	Alice and Bob common value after a successful handshake

Table 4 Notations of privacy-preserving element matching protocols.

$X = (x_1, \ldots, x_n)$	Alice's input elements
$Y = (y_1, \ldots, y_m)$	Bob's input elements
$X \cap Y = (xy_1, \ldots, xy_w)$	Intersection of Alice's and Bob's input sets
x_i, y_j	i-th and j-th elements of X and Y, respectively
k_{EA}, k_{DA}	Alice matching keys for encryption and decryption, respectively
k_{EB}, k_{DB}	Bob matching keys for encryption and decryption, respectively
$E(X) = (cx_1, \ldots, cx_n)$ $E(Y) = (cy_1, \ldots, cy_m)$	Alice's and Bob's encrypted element sets, respectively
$cx_i = cx_{i\_f}, cx_{i\_b}$	Front part and back part of Alice's i-th encrypted element
$cy_j = cy_{j\_f}, cy_{j\_b}$	Front part and back part of Alice's j-th encrypted element
$M(E(X)) = (mx_1, \ldots, mx_n)$ $M(E(Y)) = (my_1, \ldots, my_m)$	Alice's modified encrypted element set and Bob's modified encrypted element set, respectively
$mx_i = mx_{i\_f}, mx_{i\_b}$	Front part and back part of Alice's i-th modified encrypted element
$my_j = my_{j\_f}, my_{j\_b}$	Front part and back part of Alice's j-th modified encrypted element

C. GENERAL DEFINITIONS AND ADVERSARY MODEL

Fig. 1 presents a general illustration of the building elements of both PSI and SH protocols. Here, a *central authority* (CA) assigns a *credential* (e.g., VIP membership of a bank) separately to each of the parties, Alice and Bob. After the registration process, the CA is no longer involved in interactions between Alice and Bob. Parties who have the same credential are considered in a same group. Privacy-preserving authentication of credentials between parties can be done by SH protocols. If an authentication succeeds (i.e., they are in a same group), both of them know this fact. Otherwise, no information about each other's identity and affiliation (who assigns the credential) can be unveiled. Alice and Bob also have their own *elements*, and they can perform privacy-preserving matching of their elements based on a PSI protocol, without any involvement of the CA. After the execution of a PSI protocol, Alice and Bob can only know the intersection of their element sets. Alice cannot determine Bob's elements that are not in the intersection set, and vice versa. In the follows we first present the formal definition of PSI and SH protocols.

Definition II.1 Private Set Intersection Protocol. Alice has her element set $X= (x_1, \ldots, x_n)$ and Bob has his element set $Y = (y_1, \ldots, y_m)$. After the execution of a PSI protocol, Alice (or Bob or both) knows $X \cap Y$. Alice cannot know Y', where $\forall y_j \in Y'$, $y_j \in Y$ and $y_j \notin X$. Bob cannot know X', where $\forall x_i \in X'$, $x_i \in X$ and $x_i \notin Y$. A simple case of the protocol is as follows: suppose Alice has element x and Bob has element y, they want to find whether $x = y$. If yes, they know the fact. If not, Alice does not know y and Bob does not know x. More specifically, PSI protocols need to satisfy the following requirements:

Requirements of PSI Protocols include (1) $(x_1, \ldots, x_n)$and $(y_1, \ldots, y_m)$ should be exchanged in some "format", (2) If $x_i = y_j$, Alice and Bob can find out the fact. Otherwise, nothing is disclosed, (3) Messages transmitted during the matching process do not reveal any sensitive information.

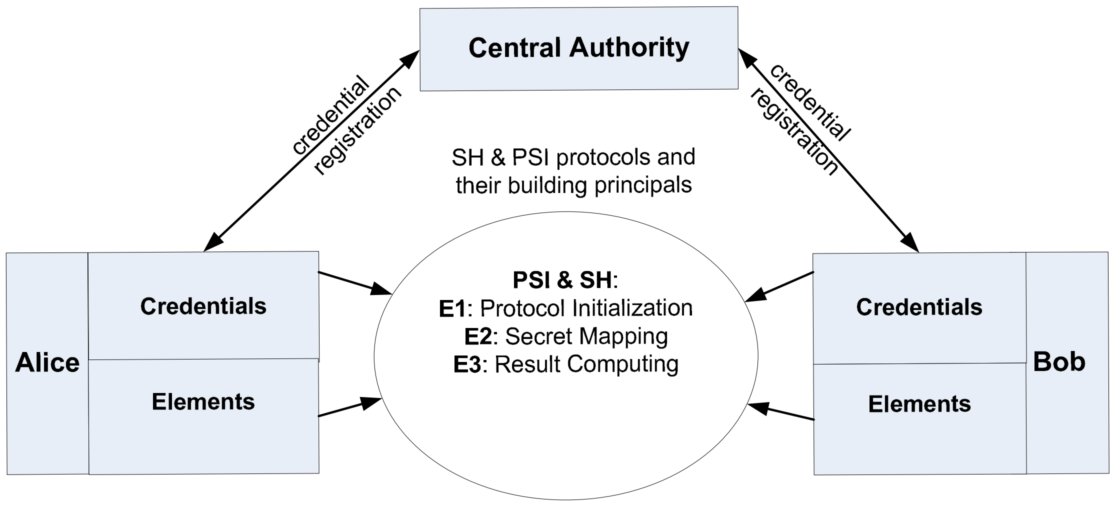


Fig. 1. A general framework of PSI and SH protocols.

Definition II.2 Secret Handshake Protocol. Alice has a pair consisting of a pseudonym and a credential, id_A and $cred_A$, assigned from a central authority. Bob has his id_B and $cred_B$ assigned from a central authority. After the execution of an SH protocol, Alice and Bob know that $cred_A$ and $cred_B$ indicate common same group membership (i.e., $cred_A$ and $cred_B$ contain a same group secret) if SH succeeds; otherwise, Alice (Bob) cannot know anything about $cred_B(cred_A)$. In addition, if Alice has two executions of a SH protocol with Bob, he cannot know that he interacted with one same party. More specifically, SH protocols need to satisfy the following requirements:

Requirements of SH Protocols include (1) id_A (id_B) should not be linked to Alice (Bob), (2) If Alice and Bob are in a same group, they can compute two identical verification values independently such that they know $cred_A$ "="$cred_B$, (3) If Alice and Bob are not in a same group, Alice (Bob) cannot derive any information (i.e., affiliation) from the verification value sent from Bob (Alice).

By looking through the formal definitions of both PSI and SH protocols, we point out that they share two requirements: (1) if and only if Alice and Bob have matching elements (credentials), the equality of the matching elements (credentials) must be computable by both principals. (2) Outsiders cannot infer any of the matching/authentication results. Bob cannot know unmatched elements/credentials of Alice, and vice versa.

In Fig. 2 we present the general adversary and attack model for privacy-preserving element matching and credential authentication protocol. In our model we assume that Alice (legitimate party) interacts with Bob (legitimate party) to execute both secret handshake protocols and private set intersection protocols. For simplicity we assume that both Alice and Bob belongs to one specific group, respectively. The SH protocols are for privacy-preserving authentication between Alice's credential set $CRED_A = f(ID_A, g_A) = (cred_{A1}, \ldots, cred_{An}) = (f(id_{A1}, g_A), \ldots, f(id_{An}, g_A))$ and Bob's credential set $CRED_B = f(ID_B, g_B) = (cred_{B1}, \ldots, cred_{Bm}) = (f(id_{B1}, g_B), \ldots, f(id_{Bm}, g_B))$. Here $ID_A = (id_{A1}, \ldots, id_{An})$ $(ID_B = (id_{B1}, \ldots, id_{Bm}))$ represents Alice's (Bob's) assigned pseudonym set, $g_A(g_B)$ represents the secret of the single group Alice (Bob) belongs to and f represents the function that associates a group secret with a pseudonym. The PSI protocols are for privacy-preserving matching of Alice's element set $X = (x_1, x_2, \ldots, x_n)$ and Bob's element set $Y = (y_1, y_2, \ldots, y_m)$.

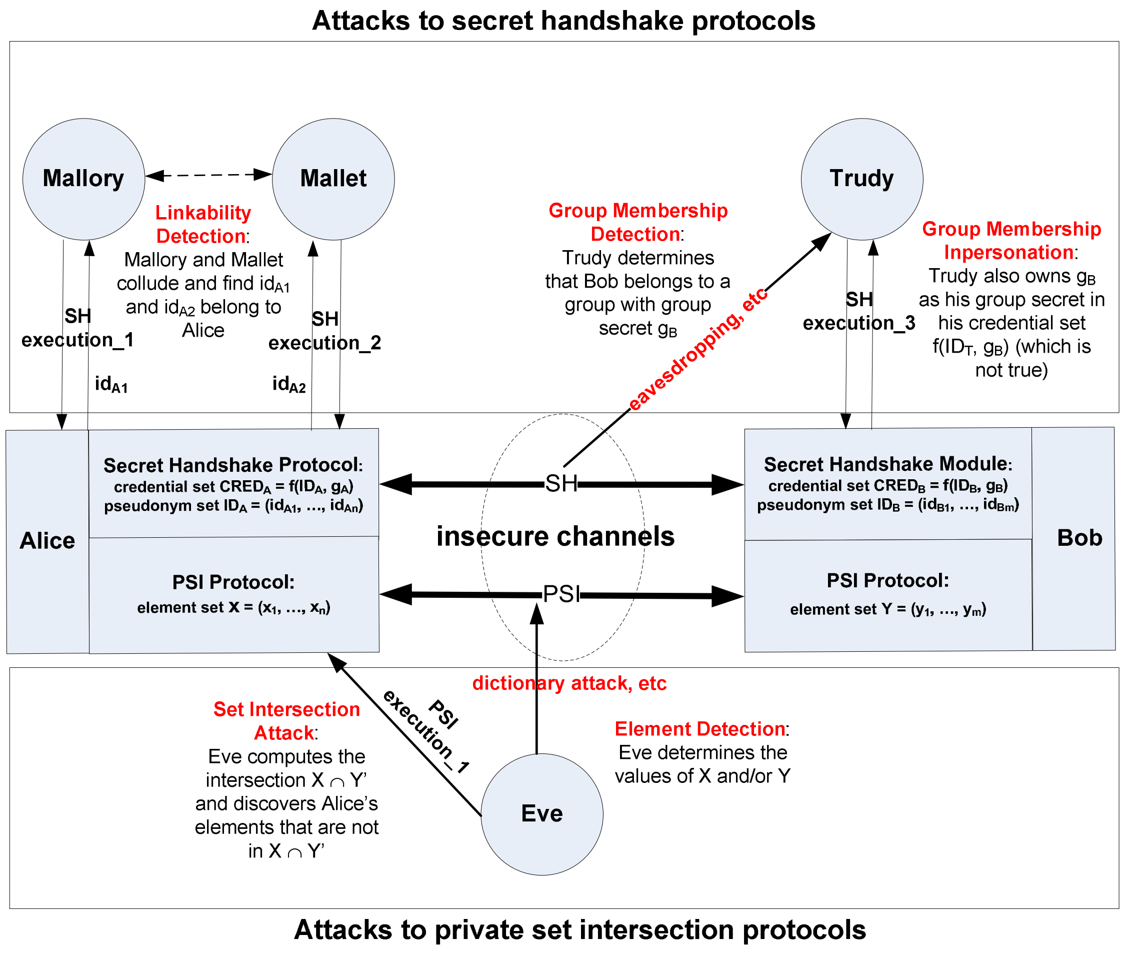

Attacks to secret handshake protocols
Mallory
Mallet
Trudy
Linkability Detection: Mallory and Mallet collude and find id_{A1} and id_{A2} belong to Alice
Group Membership Detection: Trudy determines that Bob belongs to a group with group secret g_B
Group Membership Inpersonation: Trudy also owns g_B as his group secret in his credential set $f(ID_T, g_B)$ (which is not true)
SH execution_1
id_{A1}
SH execution_2
id_{A2}
SH execution_3
eavesdropping, etc
Alice
Secret Handshake Protocol: credential set $CRED_A = f(ID_A, g_A)$ pseudonym set $ID_A = (id_{A1}, \ldots, id_{An})$
PSI Protocol: element set $X = (x_1, \ldots, x_n)$
SH
insecure channels
PSI
Secret Handshake Module: credential set $CRED_B = f(ID_B, g_B)$ pseudonym set $ID_B = (id_{B1}, \ldots, id_{Bm})$
PSI Protocol: element set $Y = (y_1, \ldots, y_m)$
Bob
dictionary attack, etc
Set Intersection Attack: Eve computes the intersection $X \cap Y'$ and discovers Alice's elements that are not in $X \cap Y'$
PSI execution_1
Element Detection: Eve determines the values of X and/or Y
Eve
Attacks to private set intersection protocols

Fig. 2. General adversary and attack model for privacy-preserving protocols.

In the model there are two kinds of adversaries: *active adversaries* Mallory, Mallet and Trudy and *passive adversary* Eve. Here for simplicity Trudy also launches passive attacks and Eve also launches active attacks.

For secret handshake protocols there are three kinds of attacks: linkability detection, group membership detection and group membership impersonation. Among them linkability detection and group membership impersonation are active attacks since adversaries need to interact with parties to launch the attacks. Group membership detection is a passive attack since adversaries can utilize eavesdropping or interception techniques to get the transmitted messages and then analyze the information to launch the attack. In the follows we describe the detail of the attack scenarios.

Definition II. 3 Linkability Detection. Mallory executes a SH protocol with Alice in SH execution_1 and Mallet also executes a SH protocol with Alice in SH execution_2. In SH execution_1 Alice sent her pseudonym id_{A1} to Mallory for computing the verification value and in SH execution_2 Alice sent her pseudonym id_{A2} to Mallet. After the executions (successful or not) Mallory and Mallet collude and try to figure out whether id_{A1} and id_{A2} belong to a same party, i.e., Alice.

Definition II. 4 Group Membership Impersonation. Trudy executes a SH protocol with Bob in SH execution_3 and try to make it successful. That said, Trudy tries to convince Bob that he is also a owner of the group secret g_B, i.e., he owns a credential in credential set $f(ID_T, g_B)$, where ID_T is his own pseudonym set.

Definition II. 5 Group Membership Detection. Trudy chooses a target group secret g_B' and tries to find out whether Bob's group secret is g_B', i.e., whether $g_B = g_B'$.

For private set intersection protocols there are two kinds of attacks: element detection and set intersection attack. Among them element detection is a passive attack since adversaries can utilize eavesdropping or interception techniques to get the transmitted messages and then analyze the information by launching dictionary attacks. Set intersection attack is an active attack since adversaries need to interact with parties to launch the attacks. In the follows we describe the detail of the attack scenarios.

Definition II. 6 Element Detection. Eve tries to find out the values of X and/or Y by launching any passive attack on Alice's and Bob's transmitted messages in their PSI execution.

Definition II. 7 Set Intersection Attack. Eve executes a PSI protocol with Alice in PSI execution_1. She computed the intersection of their elements $X \cap Y'$, where Y' represents the set of her own elements. Eve tries to find out Alice's elements that are not in $X \cap Y'$.

D. A GENERAL PRIVACY-PRESERVING FRAMEWORK

To satisfy the requirements of PSI and SH protocols and defend against the attacks mentioned above, first Alice needs to initialize a matching/authentication process by sending her parameters to Bob. Here we assume that Alice is the sender who initializes the matching/authentication protocol and Bob is the receiver who responds to Alice's request. The first phase is called *Protocol Initialization*. After Bob received the request, he needs to find a way to map his secrets (i.e., elements in PSI and credential in SH) to another type of hidden values such that two requirements are satisfied: (1) his hidden values can be used to match or authenticate his elements/credentials, (2) it is computationally infeasible to derive his real secrets from the hidden values. This is the key part of any PSI or SH protocols. Usually we use mathematic function with homomorphism to achieve the two conflicting goals simultaneously. This phase is called *Secret Mapping*. After the mapped secrets are sent back to Alice, she needs to compare the received hidden values with her own secrets. This phase is called *Result Computing*, which should satisfy two requirements: (1) Alice could compute the equality between their secrets if they have matched elements/credentials, (2) otherwise, it is computationally infeasible for Alice to deduce Bob's secrets from his hidden values. Now we propose a formal statement to illustrate the general design approach for both PSI and SH protocols.

Statement II.1 To satisfy the general requirements of both PSI and SH protocols, we propose a general approach with three common phases: *Protocol Initialization* phase, *Secret Mapping* phase and *Result Computing* phase. The general approach for Alice (sender) and Bob (receiver) is as follows: (1) Alice initializes the matching/authentication protocol by sending her parameters to Bob in Protocol Initialization phase, (2) Bob received the parameters and maps his secrets to hidden values and sends the mapped values back to Alice in Secret Mapping phase, (3) Alice received the mapped values and compute whether they have matched elements/credentials in Result Computing phase. The difference between a PSI protocol and a SH protocol is that a party can manipulate the secret (i.e., self-generated elements) in PSI protocol and cannot do so (i.e., assigned credentials) in SH protocol.

Now we present the detail of the three phases. For simplicity, we still assume that Alice is sender who initializes a matching/authentication protocol and Bob is the receiver who responds to Alice's request. The process is similar when Bob is the sender and Alice is the receiver.

Phase I Protocol Initialization

Alice sends several parameters to Bob to initialize a matching/authentication protocol. These parameters include:

P_B: the base element of a chosen group G_1, where a homomorphic operation * is defined and the inverse operation of * on G_1 is hard (e.g., DLP or ECDLP)

s_A: Alice's secret, which can be her elements, identity, pseudonym, etc.

c_A: the comparison key used to balance the equation for both PSI protocols and SH protocols. In PSI protocols c_A is a random number used to guarantee that only Alice can compute the matching result. In SH protocol c_A is a group secret of Alice's group membership (credential).

r_A: a random number used to guarantee that only Alice's secret/group secret cannot be deduced by Bob.

R: a ***homomorphic randomization function*** that associates a secret with randomly chosen numbers in certain format.

If Alice chooses to use the homomorphic randomization function and associate r_A with her secret s_A and send the mapped value to Bob, the general equation of this phase for PSI protocols is $R(r_A, c_A, s_A) = (r_A \times s_A \times c_A) * P_B$, were * represents the homomorphic operation on G_1 and $\times$ is the regular multiplication operation. For SH protocols the equation of this phase is $R(r_A, c_A, s_A) = (r_A \times s_A) * P_B$.

In this phase we propose to use R, a homomorphic randomization function, to allow Alice to associate the random numbers r_A with her secret s_A so that s_A cannot be deduced from the randomized value. The randomization of R provides unlinkability for SH protocols and defense against dictionary attack for PSI protocols. In addition, the homomorphism of R guarantees that Bob can further modify Alice's encrypted secrets, even if he does not know the chosen random number r_A and s_A.

Phase II. Secret Mapping

After receiving Alice's parameters, Bob will map his secrets to hidden values by using a secret mapping function. This phase includes the parameters and function as follows:

s_B: Bob's secret, which can be her elements, identity, pseudonym, etc.

c_B: the comparison key used to balance the equation for both PSI protocols and SH protocols. In PSI protocols c_B is a random number used to guarantee that only Alice can compute the matching result. In SH protocol c_B is a group secret of Alice's group membership (credential).

r_B: a random number used to guarantee that Bob's secret/group secret cannot be deduced by Alice.

M: a ***secret mapping*** function used to map s_B (or c_B) to a hidden value such that Alice can match/verify s_B (or c_B), and Alice cannot deduce s_B (or c_B) if matching/authentication fails. The equation of the secret mapping function for PSI protocls is $M(R(r_A, c_A, s_A), r_B, c_B, s_B) = (r_A \times r_B \times c_A \times s_A)*P_B, (r_A \times r_B \times s_B)*P_B$. The equation of the secret mapping function for SH protocol is $M(R(r_A, c_A, s_A), r_B, c_B, s_B) = [(r_A \times s_A)*P_B] \otimes [(r_B \times c_B \times s_B)*P_B], (r_B \times s_B)*P_B$.

In this phase PSI and SH have different requirements for M. For PSI protocol, since Alice (Bob) can compute and manipulate her (his) elements, the secret mapping function only consists of homomorphic operations represented by *. For SH protocols, since Alice (Bob) cannot compute or manipulate the group secret associate with her/his credentials, the secret mapping function also consists of a bilinear mapping function represented by $\otimes$. The theoretical basis is still same in Phase II for both PSI protocols and SH protocols. That said, in both kind of protocols Bob uses his random number r_B to protect his secret s_B and c_B, while he also associates his r_B with Alice's transmitted messages through homomorphic operation or bilinear mapping for further possible comparison.

Phase III. Result Computing

After receiving Bob's hidden values, Alice will compute the matching/authentication result by using a result computing function. It includes:

C: a ***result computing*** function used to compute the matching/authentication result. It enables Alice to compute the result by comparing s_A and s_B, or c_A and c_B, and at the same time it prevents Alice from deducing s_B or c_B from the ciphertext directly. The equation of the result computing function for PSI protocols is: $C(M(R(r_A, c_A, s_A), r_B, c_B, s_B)) \Rightarrow (r_A \times r_B \times c_A \times s_A)*P_B =? (r_A \times r_B \times c_A \times s_B)*P_B$, where if $s_A = s_B$, the equation is satisfied. The equation for the result computing funciton for SH is $C(M(R(r_A, c_A, s_A), r_B, c_B, s_B)) \Rightarrow [(r_A \times s_A)*PB] \otimes [(r_B \times c_B \times s_B)*PB] =? [(r_A \times c_A \times s_A)*PB] \otimes [(r_B \times s_B)*PB]$, where if $c_A = c_B$, the equation is satisfied.

In this phase Alice uses the result computing function to find out the matched elements in PSI protocol or verified credentials in SH protocols. Basically Alice needs to put her comparison key on one side of the equation to balance it except of the comparing secrets, i.e., Alice associates her comparison key c_A with Bob's encrypted ciphertexts in both PSI protocols and SH protocol. The difference between PSI protocols and SH protocols is that the comparing secrets in the former are s_A and s_B and the comparing secrets in the latter are c_A and c_B. For PSI protocol, both left side and right side have the same parameters except of the compared secrets s_A and s_B so that if $s_A = s_B$ the equation is satisfied. For SH protocols, both left side and right side have the same parameters except of the comparison secrets c_A and c_B so that if $c_A = c_B$ the equation is satisfied. During the computation, Alice cannot know anything about Bob's secret s_B and c_B if the matching or verification fails because the randomization of HRF.

III. PRIVATE SET INTERSECTION (PSI) PROTOCOLS

A. INTRODUCTION

Numerous protocols have been proposed to achieve private set intersection. The first two-party private set matching protocol [24] was proposed based on homomorphic encryption scheme and polynomial equation evaluation. Let us assume that Alice and Bob both have a set of n attribute records for matching (equivalent to the term *element* used in [24]). In [24], Alice needs to map each record to one root of a polynomial equation, encrypts the coefficients of the equation through a homomorphic encryption scheme and sends the ciphertexts to Bob. Bob then can evaluate the polynomial based on the encrypted coefficients and his own records. Then the evaluated polynomial is sent back to Alice such that she can determine the matching outcome. The process needs $O(n^2)$ homomorphic encryptions (e.g., point multiplication in ECC) for Bob to evaluate the equation [24][26]. That is, for each input x (record), Bob needs to evaluate the whole n-term polynomial equation $a_0 + a_1 \otimes x + a_2 \otimes x^2 + \ldots + a_n \otimes x^n$ through n times operation of $\otimes$, which denotes the operation of a homomorphic encryption.

The matching protocol in [24] was extended in [32] to support multiple parties and more set operations, e.g., union, cardinality of intersection, and multiplicity test. [33] further extended the polynomial-based approach by introducing a trusted third party that certified the inputs of both parties to prevent malicious active attacks, e.g., falsification. Another kind of private set intersection approach [27] was proposed based on oblivious pseudo-random function evaluation, which was more efficient than the polynomial evaluation based approaches. This approach is only secure for a relaxed adversary model.

In [34] an improved scheme based on pseudo-random function evaluation was proposed in the standard security model, in which users needed to commit their inputs before performing private set intersection. The most recent work reported in [35] also presented a linear crypto time, DDH based scheme for private set intersection that is secure against malicious adversaries.

A privacy-enhanced matchmaking protocol [36] was proposed to support forward privacy of user's identities, and matching wishes, etc. It was based on the password-authenticated key exchange protocols [37]. A threshold homomorphic cryptosystem based privacy-preserving matching protocol was proposed in [38] for set intersection and set matching with reduced computing complexity. It was based on the combination of secret sharing and homomorphic encryption. Bilinear map was also used to design privacy-preserving set intersection protocol in [39]. Another privacy-preserving union protocol was proposed in [40] for criminal investigations.

Secure multi-party computation (SMC) was first proposed in [41]. The idea of SMC was to enable a set of untrusting parties to compute certain function based on their own private inputs without revealing their private information except the common result of the function. The *Fairplay* two-party computation system proposed in [42] implemented generic secure function evaluation and solved several secure computation problems, e.g., Millionaires problem. Theoretically SMC is a general model proposed for privacy-preserving computation of any function, including computing of the intersection. However, due to their high computing and memory costs, they were not adopted for computing of intersection [23][24].

B. A PSI PROTOCOL BASED ON FULL HOMOMORPHISM

A privacy-preserving element matching protocol based on full homomorphism is proposed in this section. The protocol is built on ECC [43][44]. Different with previous work [24][27][32][35], we propose to use a homomorphic randomization function in the Phase Initialization phase to achieve privacy-preserving matching of elements. More specifically, Alice (Bob) directly takes elements as secret numbers and associates them with chosen large random numbers such that elements cannot be derived. In addition, since the random numbers are generated through a homomorphic function, Alice (Bob) can modify Bob's (Alice's) associated values for privacy-preserving matching. In comparison with [35], our scheme achieves the same computing complexity and communication complexity based on a totally different design and mathematical primitives, i.e., the homomorphism of ECC.

First we need to generate some public parameters, $(q, G_1 Z_q, H_2, P_B)$: q is a large prime number, G_1 denotes an additive cyclic group of prime order q, Z_q is a group with prime order q, H_2 is a hash function that maps a string with arbitrary length to a string with fixed length, P_B represents the base point (generator) of G_1. Here G_1 is selected in such a way that DLP [45] is assumed to be hard on it. In our design we choose G_1 as a group of points on a chosen elliptic curve, where DLP is in the form of ECDLP [45].

Definition III.1 Homomorphic Randomization Function in PSI. Suppose Alice has elements $X = (x_1, x_2, \ldots, x_n)$. Alice uses hash function H_2 to compute $(H_2(x_1), H_2(x_2), \ldots, H_2(x_n))$ and chooses a large random number $r_{A1} \in Z_q$. Alice multiplies $(H_2(x_1), H_2(x_2), \ldots, H_2(x_n)$ with r_{A1} and the base point P_B and obtains $(r_A H_2(x_1)P_B, r_A H_2(x_2)P_B, \ldots, r_A H_2(x_n)P_B$

$\in G_1$. Bob also chooses his large random number $r_{B1} \in Z_q$ and multiplies it with Alice's randomized values $(r_A r_B H_2(a_1)P_B, r_A r_B H_2(a_2)P_B ,..., r_A r_B H_2(a_n)P_B \in G_1$.

To avoid the time-consuming polynomial evaluation, we directly use large random numbers to multiply elements to generate randomized elements and associate those with a fixed ECC point to guarantee their security property based on the ECDLP [45]. For example, given an element a_i and its computed $E(x_i) = x_i r_{A1} P_B$, where r_{A1} is a large random number and P_B is a chosen ECC point, an attacker cannot use dictionary attack to confirm that x_i has been used to generate $E(x_i)$. Then Alice sends her randomized elements to Bob such that Bob could associate (based on the homomorphism of HRF) his own elements to the received messages and send the mapped elements back to Alice. Then Alice can check the equality of their elements after computing the mapped elements. That is, Alice sends $E(x_i)$ to Bob, then Bob replies with his own randomized element $E(y_j)$ with $E(x_i)$. The key idea is to create the conditions such that, all other parameters in both $E(y_j)$ and $E(x_i)$ can be canceled and/or equalized by the computing procedures, except the elements x_i and y_j. Thus, when Alice checks the equality of $E(y_j)$ and $E(x_i)'$, $E(y_j) = E(x_i)'$ if the two elements are indeed the same, otherwise $E(y_j) \neq E(x_i)'$ and Alice cannot know y_j. Bob can make the same matching through a similar procedure. Since all elements are processed in linearly encrypted format, this approach only requires a linear number of crypto operations. The flowchart of the proposed PSI protocol is depicted in Fig. 3.

Suppose Alice has her input element set $X = \{x_1, ..., x_n\}$ and Bob has his input element set $Y = \{y_1, ..., y_m\}$. Some public parameters, (q, G_1, H_1, P_B), are used in the design. Here q is a large prime number, G_1 is chosen as the *group* of points on a chosen elliptic curve with prime order q, H_1 is a hash function that maps a string of arbitrary length to a string of fixed-length, P_B is the *base point* of G_1.

To start a round of matching between Alice and Bob, first they need to generate their own secret parameters. Alice computes an elliptic curve point k_{EA} on the curve for encryption purposes using a secret number k_{DA} and the base point P_B: $k_{EA} = k_{DA}P_B$ (i.e., the standard *point multiplication* in ECC), where $k_{DA} \in Z_q$ and $k_{EA} \in G_1$. Due to the well known ECDLP, k_{DA} cannot be deduced, given k_{EA} and P_B. To guarantee that adversaries cannot find any element by trying every possibility (if the information space of elements is not very large) in a dictionary attack, Alice also needs to generate two large random integers r_{A1}, r_{A2}, $\in Z_q$ for information hiding (i.e., mapping element values to elements on a larger group). For example, an element x_i cannot be deduced from $(r_{A1}x_i)P_B$ by any passive attack because r_{A1} and r_{A2} are large enough to guarantee the ECDLP. Following a similar argument, Bob also needs to generate $k_{EB,}$ $k_{EB} = k_{DB}P_B$, and r_{B1}, r_{B2}, $\in Z_q$. Then Alice uses E to generate the ciphertexts $E(A) = [E(x_1), E(x_2) ,..., E(x_n)]$ on the (hashed) input elements X, where each element $E(x_i)$ consists of two elliptic curve points as its "front" point and "back" point. The "front" points are produced by associating Alice's chosen random numbers with base point P_B. The "back" points are produced by associating both the random numbers and Alice's records with her computed point k_{EA}, where Alice's elements cannot be deduced because of ECDLP. Alice then sends $E(X)$ to Bob. After receiving $E(X)$ Bob uses M and his own elements B to produce $M[B, E(X)]$, i.e., *modification* of $E(X)$. The modification function M allows Bob to (1) associate both his own elements *B* and one chosen random number with the "front" points of *E(X)*, and (2) associate the same random number with the "back" points of *E(A)*. Again, Bob's elements cannot be computed from the modified "back" points because of ECDLP. Then Bob sends his *M[B, E(A)]* back to Alice, who can use *D* to decrypt-and-then-evaluate the "front" and "back" points. After the decryption (i.e., eliminating the difference between the "front" points and "back" points caused by Alice's own secret number and random number) if a pair of "front" point and "back" point are equal, Alice and Bob have an identical/matched element. Otherwise, their elements are different. Bob has symmetric operations as Alice and therefore will not be further repeated here.

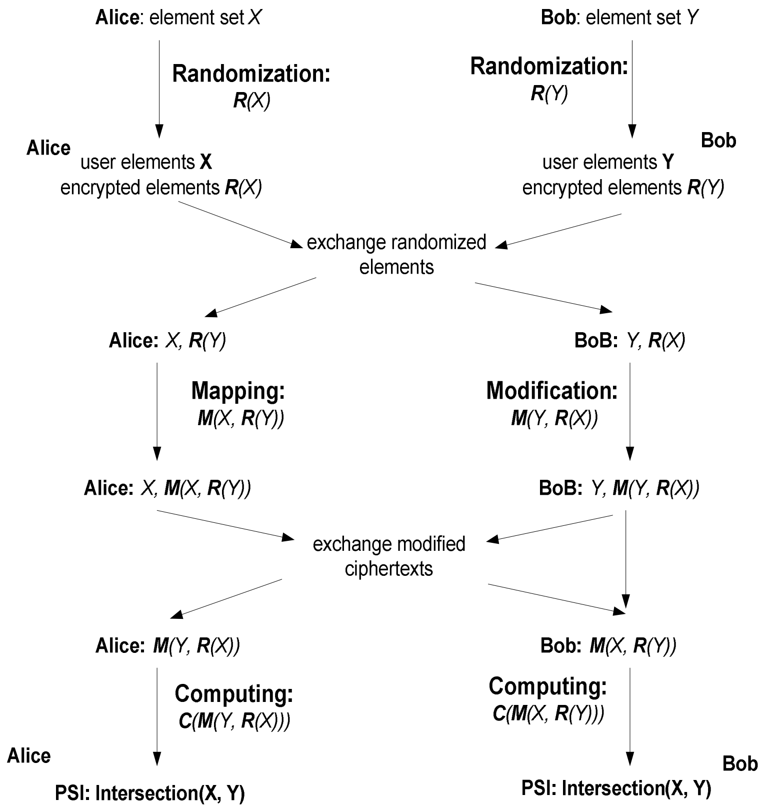


Fig. 3. The flowchart of the proposed PSI protocol.

Phase I Protocol Initialization

In the follows we only describe Alice's matching procedure since Bob's is similar. Suppose Alice has input element set $X = (x_1, x_2, \ldots, x_n)$ and Bob has input element set $Y = (y_1, y_2, \ldots, y_m)$, respectively, where n and m are two chosen positive integers. First Alice generates her pairwise matching keys (k_{EA}, k_{DA}). k_{EA} is used for encryption and k_{DA} is used for decryption. Then Alice takes the input elements $X = (x_1, x_2, \ldots, x_n)$ and uses her k_{EA} to encrypt them to get the ciphertexts $E_A(X)$. The ciphertexts are then sent to Bob. After receiving $E_A(X)$, Bob modifies Alice's ciphertexts and gets $M_B(E_A(X) = M_B(Y, E_A(X))$, where Y is Bob's input elements and $M()$ denotes a modification function with both Y and $E_A(X)$ as the inputs. Bob then sends the modified ciphertexts back to Alice. After receiving the modified encrypted elements, Alice uses the decryption function to decrypt the modified ciphertexts as $D_A(M_B(E_A(X))$. Common elements between X and Y will be discovered after the decryption.

In the follows we present the detail of the matching protocol. First Alice generates her pair of matching keys (k_{EA}, k_{DA}) based on the base element $P_B \in G_1$: $k_{EA} = k_{DA}P_B$, where $k_{DA} \in Z_q$ and $k_{EA} \in G_1$. *Bob* also generates his key pair in a similar way, i.e., $k_{EB} = k_{DB}P_B$. Before exchange their elements for matching, Alice generates two large random integers r_{A1}, $r_{A2} \in Z_q$. Bob also generates his random integers r_{B1}, $r_{B2} \in Z_q$. Alice then transfers her elements to large integers $H_2(X) = [H_2(x_1), H_2(x_2)\ldots H_2(x_n)]$ and generate her ciphertexts by using the homomorphic function as $E_A(X) = [E_A(x_1), E_A(x_2), \ldots, E_A(x_n)]$, where $\forall i \in [1, n]$, $E_A(x_i) = (E_{Af}, E_A(x_i)_b) = (r_{A1}P_B, r_{A1}H_2(a_i)k_{EA})$. *Bob* generates his ciphertexts in a similar way $E_B(Y) = [E_B(y_1), E_B(y_2), \ldots, E_B(y_m)]$, where $\forall j$

$\in [1, m]$, $E_B(y_j) = (E_{Bf}, E_B(y_j)_b) = (r_{B1}P_B, r_{B1}H_2(b_j)k_{EB})$. The detail of the privacy-preserving elements matching protocol is presented as follows:

Phase II and Phase III: Element Mapping and Result Computing

(1) Alice → Bob: $E_A(X)$

(2) Bob → Alice: $E_B(Y)$

(3) Alice: choose $r_{A2} \in Z_q$, *compute* $M_A(E_B(Y))$ *by using the homomorphic randomization function* $M_A(E_B(Y)) = [M_A(E_B(y_1)),\dots, M_A(E_B(y_m))] = [M_A(E_{Bf})_1,\dots, M_A(E_{Bf})_n)]$, $[M_A(E_B(y_1)_b), \dots, M_A(E_B(y_m)_b)]$, $\forall i \in [1, n]$, $M_A(E_{Bf})_i = r_{A2}H_2(a_i)E_{Bf} = r_{A2}r_{B1}H_2(a_i)P_B$; $\forall j \in [1, m]$, $M_A(E_B(y_j)_b) = r_{A2}E_B(y_j)_b) = r_{A2}r_{B1}H_2(b_j)k_{EB}$

(4) Bob: choose $r_{B2} \in Z_q$, *compute* $M_B(E_A(X))$ *by using the homomorphic randomization function* $M_B(E_A(X)) = [M_B(E_A(x_1)),\dots, M_B(E_A(x_n))] = [M_B(E_{Af})_1,\dots, M_B(E_{Af})_m)]$, $[M_B(E_A(x_1)_b), \dots, M_B(E_A(x_n)_b)]$, $\forall j \in [1, m]$, $M_B(E_{Af})_j = r_{B2}H_2(b_j)E_{Af} = r_{B2}r_{A1}H_2(b_j)P_B$; $\forall i \in [1, n]$, $M_B(E_A(x_i)_b) = r_{B2}E_A(x_i)_b) = r_{B2}r_{A1}H_2(a_i)k_{EA}$

(5) Alice → Bob: $M_A(E_B(Y))$

(6) Bob → Alice: $M_B(E_A(X))$

(7) Alice: for $j = 1$ *to* m, *compute* $k_{DA}M_B(E_{Af})_j$, *for* $i = 1$ *to* n, $j = 1$ *to* m, *test whether* $k_{DA}M_B(E_{Af})_j = M_B(E_A(x_i)_b)$. *If yes, put the corresponding element into* $X \cap Y$ *as the intersection set derived by Alice;*

(8) *Bob: for* $i = 1$ *to* n, *compute* $k_{DB}M_A(E_{Bf})_i$, *for* $i = 1$ *to* n, $j = 1$ *to* m, *test whether* $k_{DB}M_A(E_{Bf})_i = M_A(E_B(y_j)_b)$. *If yes, put the corresponding element into* $X \cap Y$ *as the intersection set derived by Bob;*

Protocol I significantly reduces the computing costs. If $n = m$, both Alice and Bob need $(4n + 6)$ (i.e., $O(n)$) point multiplications to compute the intersection: 1 multiplication to generate the key pair, $(n + 2)$ multiplications to generate the ciphertexts, $(2n + 2)$ multiplications to execute modification on received ciphertexts, $(n + 1)$ multiplications to execute decryption to decrypt the modified ciphertexts.

Table 5 presents the comparison of computing costs of Protocol I and the PSI protocols proposed in [24][27]. Here n denote the number of elements of both Alice and Bob, and L is the bit length of each input element (assume all elements have the same length). Through Table 5 we can find that compared to [24][27] Protocol I has linear computation costs based on same security model and different cryptography fundamentals.

Table 5. Comparison between PSI protocols.

	[24]	[27]	Protocol I	Protocol II
Computation Cost	$O(n^2)$	O(nL)	O(n)	O(n)
Communication Cost	O(n)	O(nL)	O(n)	O(n)
Security Analysis	Secure to passive attack	Secure to passive attack	Secure to passive attack	Secure to passive attack
Cryptography Fundamental	Oblivious polynomial evaluation (OPE)	Oblivious evaluation of pseudorandom function (OPRF)	ECC-based full homomorphism and ECDLP	ECC-based partial homomorphism and ECDLP

We consider any passive attack that aims at compromising confidentiality of sensitive information by analyzing transmitted messages, e.g., dictionary attack to transmitted messages. We consider both outside adversaries and inside adversaries in our research as follows.

Outside adversary: a malicious user that is an outsider of a matching process. An outside adversary does not participate in the matching process and knows nothing about the transmitted messages.

Inside adversary: a malicious user that participates in a matching process. The adversarial insider may have some matched elements with the targeted victim and try to discover other unmatched elements the victim has.

Based on the adversary model and attack model introduced above, we claim that our privacy-preserving correlation technique provides the following security properties: Private Set Intersection and Elements Confidentiality.

Theorem III.1 Element Confidentiality

In Protocol 2 a polynomial-time passive adversary cannot learn what element a party owns. In other words, any polynomial-time adversary, either outside adversary or inside adversary, only has negligible probability to know which element a party owns without compromising legitimate owners of the targeted elements.

Proof: Suppose there is an adversary Eve who aims at finding out the contents of some targeted element x. Eve may communicate with legitimate owners of the targeted element x, corrupt some valid users and obtain their elements. Here we use U^T to denote the set of users who own the targeted element. Eve picks a target user $u^T \in U^T$, and wants to find out the element by communicating with u^T. Now we define the *Element Detection Game* for a randomized, polynomial-time passive adversary A as follows:

Step 1: The adversary Eve communicates with owners of the targeted element based on its own choice. Eve may compromise certain user $U^C \subseteq U$ and obtain their element, where U^C denotes the set of compromised users and U denotes the whole user set.

Step 2: Eve selects a target user $u^T \notin U^C$ and $u^T \in U^T$, where users in U^T own the targeted element x.

Step 3: Eve generates x' by communicating with u^T.

We say that Eve wins the Element Detection Game when $x' = x$. Now we define the following probabilities $P = Pr[$Eve *wins Element Detection Game*$]$. When Eve does not compromise any valid owner of the targeted element x, the above probability becomes: $P'_{(U^C \cap U^T) = \varnothing} = Pr[$Eve *wins Element Detection Game* $\mid (U^C \cap U^T) = \varnothing]$. Now we can define *Element Confidentiality* of our protocol. Suppose Eve never compromises a valid owner of the target x. In other words, $(U^C \cap U^T) = \varnothing$. Our protocol is said to have Element Confidentiality if $P'_{(U^C \cap U^T) = \varnothing}$ is negligible for any passive adversary Eve. To prove that our protocol has the property, we first present the following theorem that has been proved in [24]: C's Privacy is preserved and S's Privacy is preserved. Now we compare our protocol with [24] and present the following corollary: If private set matching model proposed in [24] has preserved user's privacy, then our matching protocol holds Element Confidentiality. The difference between our protocol and the PSI protocol proposed in [24] is that our protocol generates elliptic curve point as the user's input ciphertexts and uses $E_A(X)$ and $E_B(Y)$ as user's ciphertexts instead of the equation ciphertexts in [24]. Because all ciphertexts are manipulated with single-use random number r_{A1}/r_{B1} and r_{A2}/r_{B2} through full ECC-based homomorphism, i.e., $E_A(x_i) = (E_{Af}, E_A(x_i)_b) = (r_{A1}P_B, r_{A1}H_2(a_i)k_{EA})$ and $E_B(y_j) = (E_{Bf}, E_B(y_j)_b) = (r_{B1}P_B, r_{B1}H_2(b_j)k_{EB})$, by the assumptions that DLP is hard in G_1, the ciphertexts hold the same security property as the ciphertexts do in [24]. Thus our protocol holds Element Confidentiality.

Theorem III.2 Private Set Intersection

In Protocol 2 matching parties only discover matched elements in the intersection set of their input elements. In other words, one party only has negligible probability to compute the other party's unmatched elements.

Proof: the proof is similar to that of Theorem III.1 and omitted here.

C. A PSI PROTOCOL BASED ON PARTIAL HOMOMORPHISM

We note that the design of Protocol I is based upon the security framework of the *private set intersection* (PSI) protocol first proposed in [24], but we have developed a much more efficient algorithm for private computing of the set intersection. We used the full homomorphic randomization function to achieve the goal of privacy-preserving element matching. However, it is vulnerable to active attack. For example, $H_2(x)P_B$ can be modified to $H_2(x')H_2(x)P_B$ by adversaries. In this section we present a linear private set intersection protocol that is based on partial homomorphic randomization function.

First we need to generate some public parameters, $(q, G_1\ Z_q, H_2, P_B)$: q is a large prime number, G_1 denotes an additive cyclic group of prime order q, Z_q is a group with prime order q, H_2 is a hash function that maps a string with arbitrary length to a string with fixed length, P_B represents the base point (generator) of G_1. Here G_1 is selected in such a way that DLP *[45]* is assumed to be hard on it.

Definition III.2 Partial Homomorphic Randomization Function in PSI. Suppose Alice has elements $X = (x_1, x_2, \ldots, x_n)$. Alice uses hash function H_2 to compute $(H_2(x_1), H_2(x_2), \ldots, H_2(x_n))$ and chooses two large random numbers $r_{A1}, r_{A2} \in Z_q$. Alice multiplies $(H_2(x_1), H_2(x_2), \ldots, H_2(x_n)$ with r_{A1} and r_{A2} and the base point P_B to obtain: $\forall i \in [1, n]$, $E(x_i)_f = r_{A1}r_{A2}^{-1}P_B \in G_1$ and $E(x_i)_b = [r_{A1}H_2(x_i) + r_{A2}]k_{EA} \in G_1$. Bob also chooses his elements $Y = (y_1, y_2, \ldots, y_m)$ and a large random number $r_{B3} \in Z_q$ and multiplies it with Alice's randomized values to obtain: $\forall j \in [1, m]$, $M(y_j)_f = H_2[r_{B3}H_2(y_j)E(x_i)_f + r_{B3}P_B]$ and $\forall i \in [1, n]$, $M(x_i)_b = r_{B3}E(x_i)_b$.

This protocol consists of three major functions: an *encryption function E*, a *decryption function D* and a *modification function M*. Let $X = (x_1, \ldots, X_n)$ and $Y = (y_1, \ldots, y_m)$ respectively represent the set of elements owned by Alice and Bob. First Alice and Bob need to generate their own secret parameters. Alice computes an elliptic curve point k_{EA} on the chosen elliptic curve for encryption purposes using a secret number k_{DA} and the base point P_B: $k_{EA} = k_{DA}P$ (i.e., the standard *point multiplication* in ECC), where $k_{DA} \in Z_q$ and $k_{EA} \in G_1$. Due to the well known ECDLP, k_{DA} cannot be deduced, given k_{EA} and P_B. To guarantee that adversaries cannot find any element by trying every possibility (the information space of alert attributes is usually not very large) in a dictionary attack, Alice also needs to generate three large random integers $r_{A1}, r_{A2}, r_{A3} \in Z_q$ for information hiding (i.e., mapping record values to elements on a larger group). For example, a element x_i cannot be deduced from $(r_{A1}x_i + r_{A2})P$ by any passive attack because r_{A1} and

r_{A2} are large enough to guarantee the ECDLP. Following a similar argument, Bob also needs to generate k_{EB}, $k_{EB} = k_{DB}P_B$, and r_{B1}, r_{B2}, $r_{B3} \in Z_q$.

Now we introduce the basic procedures of the matching scheme from Alice's side. First Alice uses E to generate the ciphertexts $E(X) = [E(x_1), E(x_2), \ldots, E(x_n)]$ on the (hashed) input elements E, where each element $E(x_i)$ consists of two elliptic curve points as its "front" point and "back" point. The "front" points are produced by associating Alice's chosen random numbers with base point P_B. The "back" points are produced by associating both the random numbers and Alice's elements with her computed point k_{EA}, where Alice's records cannot be deduced because of ECDLP. Alice then sends $E(X)$ to Bob. After receiving $E(X)$ Bob uses M and his own elements B to produce $M[Y, E(X)]$, i.e., *modification* of $E(X)$. The modification function M allows Bob to (1) associate both his own elements Y and one chosen random number with the "front" points of $E(X)$, and (2) associate the same random number with the "back" points of $E(X)$. Again, Bob's elements cannot be computed from the modified "back" points because of ECDLP. Then Bob sends his $M[Y, E(X)]$ back to Alice, who can use D to decrypt-and-then-evaluate the "front" and "back" points. After the decryption (i.e., eliminating the difference between the "front" points and "back" points caused by Alice's own secret number and random number) if a pair of "front" point and "back" point are equal, Alice and Bob have an identical/matched element. Otherwise, their records are different. Bob has symmetric operations as Alice and therefore will not be further repeated here. Details of the protocol are discussed as follows.

Protocol II. Linear Private Set Intersection Protocol Based on Partial Homomorphic Randomization Function

(1) Alice: generate $E(X) = [E(x_1), E(x_2), \ldots, E(x_n)]$, $\forall i \in [1, n]$, $E(x_i) = [E(x_i)_f, E(x_i)_b]$, *where* $E(x_i)_f = r_{A1}\, r_{A2}^{-1} P_B$, *and* $E(x_i)_b = [r_{A1}H_2(x_i) + r_{A2}]k_{EA}$

(2) Bob: generate $E(Y) == [E(y_1), E(y_2), \ldots, E(y_m)]$, $\forall j \in [1, m]$, $E(y_j) = [E(y_j)_f, E(y_j)_b]$, *where* $E(y_j)_f = r_{B1}\, r_{B2}^{-1} P_B$, *and* $E(y_j)_b = [r_{B1}H_2(y_j) + r_{B2}]k_{EB}$

(3) Alice $\leftrightarrow$ *Bob: Alice sends E(X) to Bob and Bob sends E(Y) to Alice*

(4) Alice: compute $M[X, E(Y)]$. $M[X, E(Y)] = [M(x_1)_f, M(x_2)_f, \ldots, M(x_n)_f], [M(y_1)_b, M(y_2)_b, \ldots, M(y_m)_b]$. $\forall i \in [1, n]$, $M(x_i)_f = H_1[r_{A3}H_2(x_i)(r_{B1}r_{B2}^{-1})P_B + r_{A3}P_B] = H_2[(r_{A3}r_{B1}r_{B2}^{-1}H_2(x_i) + r_{A3})P_B]$. $\forall j \in [1, m]$, $M(y_j)_b = r_{A3}[(r_{B1}H_2(y_j) + r_{B2})k_{EB}] = [r_{A3}r_{B1}H_2(y_j) + r_{A3}r_{B2}]k_{EB}$

(5) Bob: compute $M[Y, E(X)]$. $M[Y, E(X)] = [M(y_1)_f, M(y_2)_f, \ldots, M(y_m)_f], [M(x_1)_b, M(x_2)_b, \ldots, M(x_n)_b]$. $\forall j \in [1, m]$, $M(y_j)_f = H_2[r_{B3}H_1(y_j)(r_{A1}r_{A2}^{-1})P_B + r_{B3}P_B] = H_2[(r_{B3}r_{A1}r_{A2}^{-1}H_2(y_j) + r_{B3})P_B]$. $\forall i \in [1, n]$, $M(x_i)_b = r_{B3}[(r_{A1}H_2(x_i) + r_{A2})k_{EA}] = [r_{B3}r_{A1}H_2(x_i) + r_{B3}r_{A2}]k_{EA}$

(6) Alice $\leftrightarrow$ *Bob: Alice sends M[X, E(Y)] to Bob, and Bob sends M[Y, E(X)] to Alice*

(7) Alice: compute D{M[Y, E(X)]}: for i = 1 to n, compute $H_1[k_{DA}^{-1}r_{A2}^{-1}M(x_i)_b]$. *For i = 1, 2, … n, j = 1, 2, … m, test whether* $H_2[k_{DA}^{-1}r_{A2}^{-1}M(x_i)_b] = M(y_j)_f$, *if yes,* x_i *is a matched record in the intersection.*

(8) *Bob: compute D{M[X, E(Y)]}: for j = 1 to m, compute* $H_1[k_{DB}^{-1}r_{B2}^{-1}M(y_j)_b]$. *For i = 1, 2, … n, j = 1, 2, … m, test whether* $H_2[k_{DB}^{-1}r_{B2}^{-1}M(y_j)_b] = M(x_i)_f$. *If yes,* y_j *is a matched record in the intersection*

We use the multiplications of large random numbers with ECC points to protect user's elements from dictionary attacks, i.e., $H_2(x_i)$ cannot be deduced from $E(x_i)_b = [r_{A1}H_2(x_i) + r_{A2}]k_{EA}$. In addition, we also use H_2 on $M[X, E(Y)]$ and $M[Y, E(X)]$ such that Alice (or Bob) cannot further manipulate the received ciphtertexts as ECC points. Otherwise, inside adversaries, e.g., Alice, could launch dictionary attack to try every possible values for elements to get an equivalent equation between $H_2[k_A^{-1}r_{A2}^{-1}M(x_i)_b]$ and $M(y_j)_f$.

If $n = m$, both Alice and Bob need $(4n + 6)$ (i.e., O(n)) point multiplications to compute the intersection: one multiplication to associate secret number with the base point, $(n + 2)$ multiplications to generate the ciphertexts (step 1 or step 2), $(2n + 2)$ multiplications to execute M on received ciphertexts (step 5 or step 6), $(n + 1)$ multiplications to execute D on the modified ciphertexts (step 9 or step 10) for decryption.

One possible attack to Protocol II is that an attacker pretends to have the full set of elements, and sends the elements to another user for matching. An easy detection of such an attack is that when a user receives an extremely large amount of encrypted elements for matching, he can choose to not continue the matching process and does not send his modified elements back to the requester.

Protocol II significantly reduces the computing costs. If $n = m$, both Alice and Bob need $(4n + 6)$ (i.e., $O(n)$) point multiplications to compute the intersection: 1 multiplication to generate the key pair, $(n + 2)$ multiplications to generate the ciphertexts, *(2n + 2)* multiplications to execute modification on received ciphertexts, *(n + 1)* multiplications to execute decryption to decrypt the modified ciphertexts.

Table 5 above presents the comparison of computing costs of Protocol II and the PSI protocols proposed in [24][27] and Protocol I. Here n denote the number of elements of both Alice and Bob, and L is the bit length of each input element (assume all elements have the same length). Through Table 5 we can find that compared to [24][27] Protocol II has linear computation costs based on same security model and different cryptography fundamentals. In addition, compared with Protocol I, Protocol II is based on partial homomorphism, which is more secure against some kind of active attacks.

In this section we show that Protocol II is more secure against active attacks like falsification. First we can see that Protocol I is vulnerable to active attack. For example, $H_2(x)P_B$ can be modified to $H_2(x')H_2(x)P_B$ by adversaries. If $H_2(x')H_2(x)P_B$ happens represent another element, i.e., $H_2(x')H_2(x) = H_2(y)$, the participating party will believe that there is a party who owns y. It is a falsification attack. Protocol II is not vulnerable to this attack since all elements are encrypted in format like $E(x_i) = [E(x_i)_f, E(x_i)_b]$, where $E(x_i)_f = r_{A1}r_{A2}^{-1}P_B$, and $E(x_i)_b = [r_{A1}H_2(x_i) + r_{A2}]k_{EA}$. It is hard for an adversary to manipulate $r_{A1}r_{A2}^{-1}P_B$ or $[r_{A1}H_2(x_i) + r_{A2}]k_{EA}$ since he does not know both r_{A1}, r_{A2} and k_{EA}. For example, if an adversary adds a chosen element to the ciphertext like $H_2(x')[r_{A1}H_2(x_i) + r_{A2}]k_{EA}$, the participating parties cannot verify the modified ciphertext since it is not under the correct format. Thus, Protocol II is more secure against active attacks.

In Protocol II we also consider any passive attack that aims at compromising confidentiality of sensitive information by analyzing transmitted messages, e.g., dictionary attack to transmitted messages. We consider both outside adversaries and inside adversaries in our research as follows. The main discussions of the adversaries are same as that for Protocol I and are omitted here.

Based on the adversary model and attack model introduced above, we claim that our privacy-preserving correlation technique provides the following security properties: Private Set Intersection and Elements Confidentiality.

Theorem III.3. Element Confidentiality

In Protocol II a polynomial-time passive adversary cannot learn what element a party owns. In other words, any polynomial-time adversary, either outside adversary or inside adversary, only has negligible probability to know which element a party owns without compromising legitimate owners of the targeted elements.

Proof: Suppose there is an adversary Eve who aims at finding out the contents of some targeted element *x*. Eve may communicate with legitimate owners of the targeted element *x*, corrupt some valid users and obtain their elements. Here we use U^T to denote the set of users who own the targeted element. Eve picks a target user $u^T \in U^T$, and wants to find out the element by communicating with u^T. Now we define the *Element Detection Game* for a randomized, polynomial-time passive adversary *A* as follows:

Step 1: The adversary Eve communicates with owners of the targeted element based on its own choice. Eve may compromise certain user $U^C \subseteq U$ and obtain their element, where U^C denotes the set of compromised users and *U* denotes the whole user set.

Step 2: Eve selects a target user $u^T \notin U^C$ and $u^T \in U^T$, where users in U^T own the targeted element *x*.

Step 3: Eve generates x' by communicating with u^T. We say that Eve wins the Element Detection Game when $x' = x$. Now we define the following probabilities $P = Pr[$Eve *wins Element Detection Game]*. When Eve does not compromise any valid owner of the targeted element x, the above probability becomes $P'_{(U^C \cap U^T)=\varnothing} = Pr[$Eve *wins Element Detection Game* | $(U^C \cap U^T) = \varnothing]$. Now we can define *Element Confidentiality* of our protocol. Suppose Eve never compromises a valid owner of the target x. In other words, $(U^C \cap U^T) = \varnothing$. Our protocol is said to have Element Confidentiality if $P'_{(U^C \cap U^T)=\varnothing}$ is negligible for any passive adversary Eve. To prove that our protocol has the property, we first present the following theorem that has been proved in [24]: C's Privacy is preserved and S's Privacy is preserved. Now we compare our protocol with [24] and present the following theorem corollary*:* If private set matching model proposed in [24] has preserved user's privacy, then our matching protocol holds Element Confidentiality. The difference between our protocol and the PSI protocol proposed in [24] is that our protocol generates elliptic curve point as the user's input ciphertexts and uses $E_A(X)$ and $E_B(Y)$ as user's ciphertexts instead of the equation ciphertexts in [24]. Because all ciphertexts are manipulated with single-use random number r_{A1}/r_{B1} and r_{A2}/r_{B2} through ECC-based partial homomorphism, i.e., $E(x_i) = [E(x_i)_f, E(x_i)_b]$, where $E(x_i)_f = r_{A1}r_{A2}^{-1}P_B$, and $E(x_i)_b = [r_{A1}H_2(x_i) + r_{A2}]k_{EA}$ and $E(y_j) = [E(y_j)_f, E(y_j)_b]$, where $E(y_j)_f = r_{B1}\, r_{B2}^{-1}P_B$, and $E(y_j)_b = [r_{B1}H_2(y_j) + r_{B2}]k_{EB}$, by the assumptions that DLP is hard in G_1, the ciphertexts hold the same security property as the ciphertexts do in [24]. Thus our protocol holds Element Confidentiality.

Theorem III.4. Private Set Intersection

In Protocol 2 matching parties only discover matched elements in the intersection set of their input elements. In other words, one party only has negligible probability to compute the other party's unmatched elements.

Proof: the proof is similar to that of Theorem III.3 and omitted here.

IV. SECRET HANDSHAKE (SH) PROTOCOLS

A. INTRODUCTION

The original secret handshake protocol [25] was proposed to achieve privacy-preserving credential authentication between two parties. The main disadvantage of their protocol is that credentials assigned from the CA are single-use and each party needs to withdraw a large number of credentials to keep their identities unlinkable. The reason of this weak point is that pseudonyms and group secret are combined to consist of credentials and are separated for use in privacy-preserving authentication process [25]. More specifically, every credential assigned from the CA has two components, a pseudonym and a group secret, associated together. Since the group secret cannot be separated from the credential (i.e., a regular party cannot know the value of the group secret), pseudonyms are exchanged in plaintext for authentication of the credentials. This actually breaks a user's unlinkability and may even compromise the group membership detection resistance since adversaries can always track a repeatedly used pseudonym to get to know who the user is and which group he belongs to, i.e., using pseudonyms separately compromises the security of the group secret. In the follows we propose a formal theorem to illustrate the vulnerability:

Theorem IV.1 Linkability with Reusable Credential

A privacy-preserving authentication protocol that uses one plaintext pseudonym for multiple authentication sessions does not hold anonymity or group membership detection resistance.

Proof: Suppose Alice requests a CA to join the group g_A. The CA grants the group membership to Alice by issuing a credential $cred_{Ai} = f(g_A, id_{Ai})$, where g_A represents the group secret and id_{Ai} represents Alice's assigned pseudonym. Alice cannot deduce g_A from $f(g_A, id_{Ai})$. There are two adversaries Mallory and Mallet who want to know whether they are interacting with same party and which affiliation this party belongs to. Assume that Mallory and Mallet are not members in group g_A. Now we define an identity detection game as follows.

(1) Mallory sends a message to Alice and Mallet sends a message to Alice. Each of them sends the message to request for an authentication session.

(2) Alice responds with his assigned pseudonym id_{Ai} to both Mallory and Mallet.

(3) Mallory communicates with Mallet and they both know that they are interacting with a same user with pseudonym id_{Ai} in two executions of the authentication protocol.

(4) Alice is linkable.

Now we assume that Mallet is also a member of group g_A and we define a group membership detection game as follows.

(1) Mallory sends a message to Alice and Mallet sends a message to Alice. Each of them sends the message to request for an authentication session.

(2) Alice responds with his assigned pseudonym id_{Ai} to both Mallory and Mallet.

(3) Mallory completes an execution of the authentication protocol with Alice and knows nothing about Alice. Mallet also completes an execution of the authentication protocol with Alice and knows that Alice is also a member of g_A.

(4) Mallet communicates with Mallory and they both know that they are interacting with a group member of g_A with pseudonym id_{Ai}.

(5) Alice is linkable and confidentiality of her group membership is compromised.

There is more related work proposed in this area. An unlinkable SH scheme with reusable credentials is proposed by Ateniese *et al*. [28], which has stronger security properties than ours since it is proved to be secure in the standard model. On the other hand, our scheme needs less computing time, based on the random oracle mode. Experimental results [28] show that their scheme needs 0.78 seconds on average to complete an unlinkable handshake, including the key generation and credential verification. They also adopted the abstract model in [24] to achieve *approximate* attribute-based matching using each round of authentication as a matching exercise. As such, their algorithm is more close to a CA based privacy management model, because the CA knows all user attributes in order to issue credentials for users to engage in matching.

A recent related work in [46] was the first one that achieves unlinkability without relying on single-use certificates and it supports revocation. But their work does not support interlocked authentication and matching for unequal sets of attributes and in their scheme a user's sensitive information must be revealed to a CA.

B. AN UNLINKABLE SH PROTOCOL WITH REUSABLE CREDENTIAL

In this section, we present our secret handshake protocol with reusable credentials. Our main idea is to use the homomorphic randomization function in the Protocol Randomization phase to randomize user's assigned pseudonyms to provide reusability of credentials. More specifically, we let a user generate a secret random number in execution of the secret handshake. The user multiplies the random number to an elliptic curve point that represents the user's pseudonym. The random number minimizes the correlation among authentication messages even they are produced by reuse of the same credential. First we present the parameters that are required in the secret handshake protocol as $(q, G_1, G_2, e, H_1, H_2)$: q is a large prime number, G_1 denotes an additive cyclic group of prime order q, G_2 denotes a multiplicative cyclic group of the same order q, H_1 is a collision-free hash function that maps a string with arbitrary length to an element in G_1, H_2 is a collision-free hash function that maps a string with arbitrary length to a string with fixed length, and e denotes a bilinear map. G_1 and G_2 are selected in such a way that DLP [45] is assumed to be hard in both of them. The bilinear map e in our protocol is a pairing that satisfies the following conditions:

Definition VI.1 Bilinear Pairing. A pairing is a bilinear map $e: G_1 \times G_1 \rightarrow G_2$ if, for any $P, Q \in G_1$ and any $a, b \in Z^*_q$ we have $e(a{\cdot}P, b{\cdot}Q) = e(a{\cdot}P, Q)^b = e(P, b{\cdot}Q)^a = e(P, Q)^{a{\cdot}b}$ *and* $e(P,Q) = e(Q, P)$

To provide efficient computation of the bilinear map, we choose G_1, G_2 and e as a set of points on an elliptic curve, a multiplicative cyclic group over integers and Tate pairing, respectively. There is a special discrete logarithm problem, Elliptic Curve Discrete Logarithm Problem (ECDLP) [43] [44], defined as the basis of elliptic curve cryptosystem (ECC) [43][44]. Based on the parameters chosen above, another assumption we need is the *BDH Assumption* described as follows:

Definition VI.2 Bilinear Diffie-Hellman (BDH) Assumption [45]. Given P, aP, bP, cP for random a, b, $c \in Z^*_q$ and $P \in G_1$, it is not possible to compute $e(P, P)^{abc}$ with a non-negligible probability, i.e., it is hard to compute $e(P, P)^{abc}$

In the follows we present the detail of the use of the homomorphic randomization in our proposed secret handshake protocol.

Definition VI.3 Homomorphic Randomization Function in SH. Suppose user Alice joins a group g_A by obtaining a pair of pseudonym id_{Ai} and credential $cred_{Ai} = f(g_A, id_{Ai}) = g_A \cdot H_1(id_{Ai})$. Here $g_A \in G_2$ represents the group secret, id_{Ai} is an arbitrary string that represents Alice's assigned pseudonym, H_1 is a function that maps a string to an element in G_1 and f is the addition operation defined in G_1. Alice selects a large random integer $r_{A1} \in G_2$, and uses the addition operation defined in G_1 to compute $rd_{Ai} = r_{A1}H_1(id_{Ai})$ (i.e., r_{A1} times of addition on $g_AH_1(id_{Ai})$).

In my design there is a central authority (CA) that assigns credentials to two principals, Alice and Bob. Alice has her identity ID_A and Bob has his identity ID_B. The CA is also responsible for generating the parameters $(q, G_1, G_2, e, H_1, H_2)$, where q denotes a large prime number, G_1 denotes the *group* of points on a chosen elliptic curve with prime order q, G_2 denotes a multiplicative cyclic group of order q, e denotes a bilinear map: $G_1 \times G_1 \rightarrow G_2$, H_1 maps a string with arbitrary length to an element in G_1, H_2 maps a string with arbitrary length to a string with fixed length. The CA also generates a set of group secrets $(g_1,\ldots, g_n)$, $\forall g_i \in Z^*_q$, g_i represents the membership of a group. The main idea of the SH protocol is to let a principal generate a secret large random integer in every secret handshake. The principal multiplies the random number to an elliptic curve point that represents his identity. The random number minimizes the correlation among authentication messages even they are produced by reusing the same credential. The flowchart of the proposed SH protocol is presented in Fig. 4.

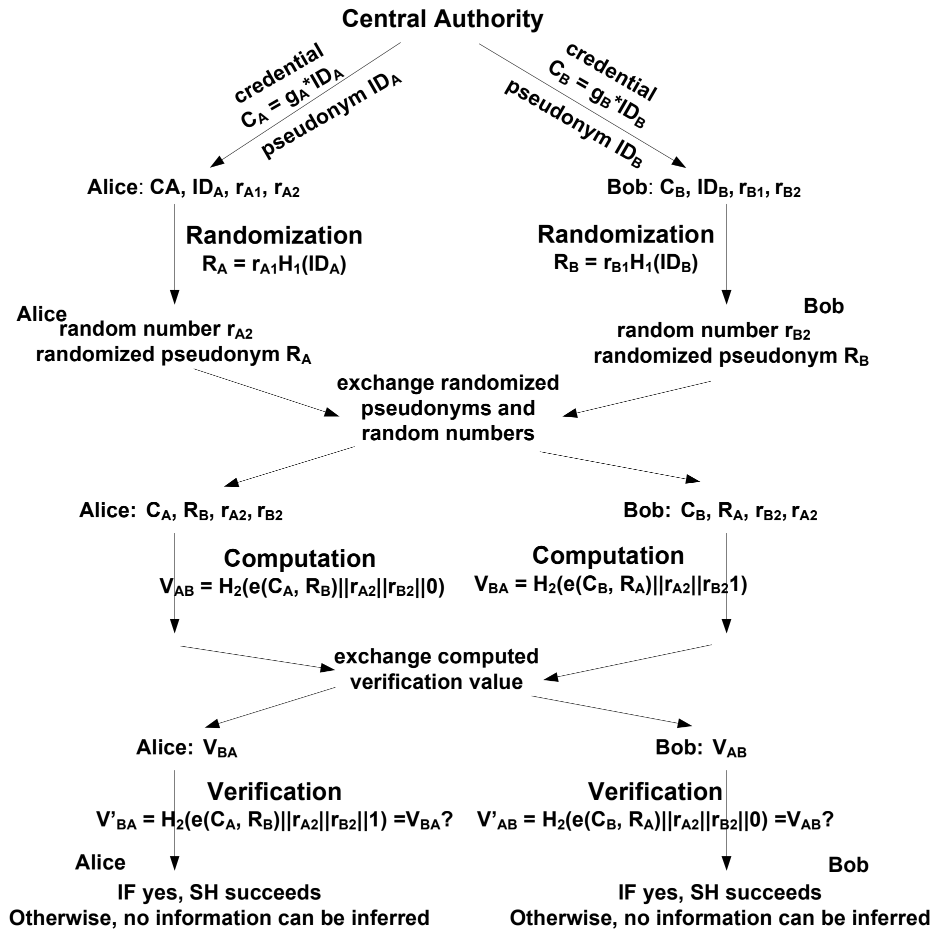

Central Authority
credential $C_A = g_A{}^*ID_A$
pseudonym ID_A
credential $C_B = g_B{}^*ID_B$
pseudonym ID_B
Alice: CA, ID_A, r_{A1}, r_{A2}
Bob: C_B, ID_B, r_{B1}, r_{B2}
Randomization
$R_A = r_{A1}H_1(ID_A)$
Randomization
$R_B = r_{B1}H_1(ID_B)$
Alice
random number r_{A2}
randomized pseudonym R_A
Bob
random number r_{B2}
randomized pseudonym R_B
exchange randomized pseudonyms and random numbers
Alice: C_A, R_B, r_{A2}, r_{B2}
Bob: C_B, R_A, r_{B2}, r_{A2}
Computation
$V_{AB} = H_2(e(C_A, R_B)||r_{A2}||r_{B2}||0)$
Computation
$V_{BA} = H_2(e(C_B, R_A)||r_{A2}||r_{B2}1)$
exchange computed verification value
Alice: V_{BA}
Bob: V_{AB}
Verification
$V'_{BA} = H_2(e(C_A, R_B)||r_{A2}||r_{B2}||1) = V_{BA}$?
Verification
$V'_{AB} = H_2(e(C_B, R_A)||r_{A2}||r_{B2}||0) = V_{AB}$?
Alice
IF yes, SH succeeds
Otherwise, no information can be inferred
Bob
IF yes, SH succeeds
Otherwise, no information can be inferred

Fig. 4. The flowchart of the proposed SH protocol.

As described in the proposed framework, our proposed secret handshake protocol has three phases: *Protocol Initialization*, *Group Secret Mapping and Authentication Computing*. In the follows we present the detail of the three phases.

Protocol III. Unlinkable Secret Handshake Protocol with Reusable Credential

Phase I Protocol Initialization

The CA determines the pairing parameters $(q, G_1, G_2, e, H_1, H_2)$ and group secrets $GS = (g_1,\ldots, g_w)$, where $\forall g_i \in G_2$. The CA publishes the pairing parameters while keeping the group secrets in private. Alice requests the CA to join the group with group secret $g_A \in (g_1,\ldots, g_w)$. The CA verifies Alice's qualification to decide whether Alice can join the group. If yes, the CA grants the group membership to Alice by issuing her a pair of pseudonym id_{Ai} and credential $cred_{Ai} = g_A H_1(id_{Ai}) \in G_1$, as described above. The credential is a secret of Alice to prove her membership in group g_A to another user in the same group. Alice cannot deduce g_A from $g_A H_1(id_{Ai})$ and $H_1(id_{Ai})$ by the assumption that DLP is hard in G_1. It is important for preventing forgery of credentials. Bob also obtained his pair of pseudonym id_{Bj} and credential $cred_{Bj} = g_B H_1(id_{Bj})$ in a similar process. Users Alice and Bob use their credentials, $cred_{Ai} = g_A H_1(id_{Ai})$ and $cred_{Bj} = g_B H_1(id_{Bj})$, to generate authentication messages to one another. Alice randomly generates two large random numbers, r_{A1} and r_{A2}. Alice computes $rd_{Ai} = r_{A1} H_1(id_{Ai})$, as described in the homomorphic randomization function above. That said, r_{A1} is used to minimize the correlations of authentication messages produced by the same pair of pseudonym and credential. Since using a credential multiple times will not create messages that can link to the user identity, our credential is reusable. r_{A2} prevents replay attacks as described in [6]. Bob also randomly generates two large numbers, r_{B1} and r_{B2}, for the same purpose. Detailed interactions of group secret mapping phase and authentication computing phase are described as follows:

Phase II Secret Mapping and Phase III Result Computing

(1) Alice →Bob: r_{A2}, $rd_{Ai} = r_{A1}H_1(id_{Ai})$

(2) Bob: Compute $V_B = H_2(e(rd_{Ai}, r_{B1}cred_{Bj}) \| r_{A2} \| r_{B2} \| 0)$*. Here Bob implements the secret mapping function to map his own credential* $cred_{Bj} = g_BH_1(id_{Bj})$ *from an element in* G_1 *to an element in* G_2 *through bilinear map e.*

(3) Bob →Alice: r_{B2}, $rd_{Bj} = r_{B1}H_1(id_{Bj})$, V_B

(4) Alice: Compute $V_{B(A)} = H_2(e(rd_{Bj}, r_{A1}cred_{Ai}) \| r_{A2} \| r_{B2} \| 0)$*. If* $V_{B(A)} = V_B$*, then Alice knows Bob belongs to the same group, i.e.,* $g_A = g_B$*. Otherwise, Bob belongs to a different group, i.e.,* $g_A \neq g_B$ *or Bob belongs to no group. Here Alice implements the result computing function to find out whether* $H_2(e(rd_{Bj}, r_{A1}cred_{Ai}) \| n_{A1} \| n_{B1} \| 0) = H_2(H_2(e(rd_{Ai}, r_{B1}cred_{Bj}) \| n_{A1} \| n_{B1} \| 0)$.

(5) Alice →Bob: $V_A = H_2(e(rd_{Bj}, r_{A1}cred_{Ai}) \| r_{A2} \| r_{B2} \| 1)$

(6) *Bob: Compute* $V_{A(B)} = H_2(e(rd_{Ai}, r_{B1}cred_{Bj}) \| r_{A2} \| r_{B2} \| 1)$*. If* $V_{A(B)} = V_A$*, Bob knows that Alice belongs to the same group. Otherwise, Alice belongs to a different group, i.e.,* $g_A \neq g_B$ *or Alice belongs to no group.*

The protocol succeeds when $V_{B(A)} = V_B$ and $V_{A(B)} = V_A$ in steps (d) and (f). Based on the BDH assumption [7], it succeeds if, and only if, $g_A = g_B$. Otherwise, if it fails, Alice and Bob only know $g_A \neq g_B$. Users other than Alice and Bob cannot know whether $g_A = g_B$ or not, because they cannot compute V_A or V_B without $cred_{Ai}$ and $cred_{Bj}$. A sketch of proof for $V_{B(A)} = V_B$ when $g_A = g_B$ is shown in (1). The rest of proof for $V_{A(B)} = V_A$ can be derived similarly.

$V_{B(A)} = H_2(e(rd_{Bj}, r_{A1}cred_{Ai}) \| r_{A2} \| r_{B2} \| 0)$

$= H_2(e(r_{B1}H_1(id_{Bj}), r_{A1}g_A \cdot H_1(id_{Ai})) \,||\, r_{A2} \,||\, r_{B2} \,||\, 0)$

$= H_2(e(H_1(id_{Bj}), H_1(id_{Ai}))^{rA1rB1gA} \,||\, r_{A2} \,||\, r_{B2} \,||\, 0)$

$= H_2(e(r_{A1}H_1(id_{Ai}), r_{B1}g_BH_1(id_{Bj})) \,||\, r_{A2} \,||\, r_{B2} \,||\, 0)$ (if $g_A = g_B$)

$= H_2(e(rd_{Ai}, r_{B1}cred_{Bj}) \,||\, r_{A2} \,||\, r_{B2} \,||\, 0)$

$= V_B$ (3. 1)

Table 6 show the comparison of computation costs and communication costs between Protocol II and the SH protocols proposed in [25][28]. Here *n* denotes the number of runs of a SH protocol and *d* denotes the threshold of the number of overlapped elements defined by Alice and Bob. My solution has less computation cost compared to [28] and less communication costs compared to [25], under different security model.

Table 6. Comparison between SH protocols.

	[25]	[28]	Protocol III
Computation Cost	O(1)	O(2d + 1)	O(1)
Communication Cost	O(n)	O(2d + 1)	O(1)
Security Analysis	Secure to passive and active attack	Secure to and active passive attack	Secure to and active passive attack
Cryptography Fundamental	Random oracle model	Standard model	ECC-based homomorphism and random oracle model

In the proposed secret handshake protocol we consider any passive attack that aims at compromising confidentiality of sensitive information by analyzing transmitted messages, e.g., dictionary attack to transmitted messages. We consider both outside adversaries and inside adversaries as follows.

Outside adversary: a malicious party that is an outsider of an authentication process. An outside adversary does not participate in the authentication process and knows nothing about the transmitted messages.

Inside adversary: a malicious party that participates in an authentication process. The adversarial insider may have some matched group secrets with the targeted victim and try to discover other unmatched group secrets the victim has.

Based on the adversary model and attack model introduced above, we claim that our privacy-preserving correlation technique provides the following main security properties: unlinkability with reusable credential, group membership authenticity, group membership detection resistance. In Theorem IV.2 we first prove that a secret handshake that uses homormorphic randomization function provides unlinkability with reusable credential.

Theorem IV.2: **Unlinkability on Homomorphic Randomization with Reusable Credential**

A privacy-preserving authentication protocol that uses homomorphic randomization function in protocol initialization phase guarantees unlinkability with reusable credential.

Proof: Suppose Alice requests to join a group with group secret $g_A \in G_2$. The CA grants the group membership to Alice by issuing a credential $g_A \cdot H_1(id_{Ai}) \in G_1$, where id_{Ai}

represents Alice's assigned pseudonym associated with g_A. Alice cannot deduce g_A from $g_A \cdot H_1(id_{Ai})$. There are two adversaries, Mallory and Mallet, who want to know whether they are interacting with same party and which group this party belongs to. Assume that Mallory is not a member in group g_A and Mallet is. Mallory sends a message to Alice to request an execution of Protocol III. Mallet also sends the request. Alice executes the protocol with both Mallory and Mallet. In the execution with Mallory Alice uses $rd_{Ai} = r_{A1} \cdot H_1 \cdot (id_{Ai})$ as her pseudonym, where r_{A1} is a randomly chosen number in group G_2. In the execution with Mallet Alice uses $rd_{Ai}' = r_{A1}' \cdot H_1 \cdot (id_{Ai})$ as her pseudonym, where r_{A1}' is another randomly chosen number in group G_2. All other parts of the executions are same as what has been described in Protocol III. Mallet may also communicate with other legitimate owners of group secret g_A. Here we use U^A to denote the set of users who own the group secret g_A. Now we define a *Linkability Detection Game* as follows.

Step 1: Mallory and Mallet communicate with Alice based on their own choices. From Mallory's viewpoint, he is interacting with party A_1; from Mallet's viewpoint, he is interacting with party A_2.

Step 2: Mallet selects other parties U^A and corrupt them.

Step 3: After the executions of the secret handshake protocol Mallory and Mallet want to find out whether $A_1 = A_2$. If yes, they know that they are interacting with a same party and since Mallet knows that A_2 has group secret g_A, Mallory also knows.

We say that Mallory and Mallet win the Linkability Detection Game if they find out that $A_1 = A_2$. Now we define the following probabilities: *P = Pr[Mallory and Mallet win Linkability Detection Game] – 0.5.* Here we say that our protocol holds unlinkability with reusable credential if *P is* negligible for any polynomial-time adversaries Mallory and Mallet. In our protocol Alice generates random numbers r_{A1} and r_{A1}' to manipulate his assigned pseudonym id_{Ai} and get elliptic curve points $rd_{Ai} = r_{A1} \cdot H_1(id_{Ai})$ and $rd_{Ai}' = r_{A1}' \cdot H_1(id_{Ai})$ as his randomized pseudonyms for Mallory and Mallet, respectively. Here rd_{Ai} and rd_{Ai}' are generated to map Alice's pseudonym id_{Ai} to random ECC points through the homomorphic randomization function. By the assumptions that DLP is hard in G_1, no polynomial-time adversary cannot deduce r_{A1} and r_{A1}' from $rd_{Ai} = r_{A1} \cdot H_1(id_{Ai})$ and $rd_{Ai}' = r_{A1}' \cdot H_1(id_{Ai})$ and thus cannot know rd_{Ai} and rd_{Ai}' are generated based on the same pseudonym id_{Ai}. That said, their guess about whether $A_1 = A_2$ is no better than a random guess. Thus, our protocol holds unlinability with reusable credential.

Theorem IV.3: Group Membership Authenticity

In Protocol III any polynomial-time adversary only has negligible probability of cheating as a valid owner of some group membership without corrupting another valid owner of the targeted group secret, i.e., the adversary cannot impersonate owners of some targeted group secret.

Proof: suppose there is an adversary Trudy who aims at impersonating the owner of a group secret g_A. Trudy may communicate with legitimate owners of the targeted g_A, corrupt some valid users and obtain their secrets. Here we use U^T to denote the set of users who own the targeted g_A. Trudy picks a target user $u^T \in U^T$, and wants to convince u^T that Trudy is also an owner of the targeted g_A, i.e., Trudy $\in U^T$. We define the *Group Membership Owner Impersonation Game* for a randomized, polynomial-time adversary Trudy as follows.

Step 1: The adversary Trudy communicates with owners of the targeted g_A based on its own choice. Trudy may compromise users $U^C \subseteq U$ and obtain their secrets, where U^C denotes the set of compromised users and U denotes the whole user set.

Step 2: Trudy selects a target user $u^T \notin U^C$ and $u^T \in U^T$, where users in U^T own the targeted g_A.

Step 3: Trudy wants to convince u^T that Trudy owns the g_A, i.e., Trudy $\in U^T$.

We say that Trudy wins the Group Membership Owner Impersonation Game if it convinces u^T that it is an owner of g_A. Now we define the following probabilities: P = *Pr[*Trudy *wins Group Membership Owner Impersonation Game]*. When Trudy does not compromise any valid owner of g_A, the above probability becomes: $P'_{(U^C \cap U^T)=\varnothing}$ = *Pr[*Trudy *wins Group Membership Owner Impersonation Game* | $(U^C \cap U^T)=\varnothing$*]*. Here we say that our protocol holds *Group Membership Authenticity* if $P'_{(U^C \cap U^T)=\varnothing}$ is negligible for any polynomial-time adversary Trudy. We first present the following theorem that has been proved in [25]: C's Authenticity is preserved and S's Authenticity is preserved. In this work we define $P'_{U^C \cap U^T)=\varnothing}$ as the probability when an adversary Trudy wins the Group Membership Owner Impersonation Game without compromising any valid owner of g_A in U^T. Now we compare our protocol with the abstract model [25] and present the following corollary: Our secret handshake protocol holds Group Membership Authenticity. The difference between our protocol and the abstract model [25] is that our protocol generates elliptic curve point as the user's randomized identities. Here rd_{Ai}/rd_{Bj} are generated to map Alice's/Bob's pseudonyms to random ECC points. Since all pseudonyms are manipulated with single-use random number r_{A1}/r_{B1} as random oracles and DLP is hard in G_1, group secret and user's pseudonyms are not deducible from the randomized values. In addition, our protocol holds the original BDH assumption that any polynomial-time adversary cannot derive the authentication value and group secret from the transmitted messages. As a result, Group Membership Authenticity holds in our secret handshake protocol.

Theorem IV.4: Group Membership Detection Resistance

In Protocol III any polynomial-time adversary only has negligible probability of detecting a targeted party's group secret without corrupting another valid owner with the same group secret. In other words, given an arbitrary group secret, the adversary's guess about whether it is same as a targeted party's group secret is no better than a random guess.

Proof: the proof is similar to Theorem IV.3 and omitted here.

V. INTERLOCKED PRIVACY-PRESERVING PROTOCOLS

A. INTRODUCTION

In this section we present two new privacy-preserving protocols by interlocking the proposed PSI protocol and the proposed SH protocol. In the first interlocked protocol a PSI protocol is executed first and the matched elements are obtained. Then the matched elements are associated with an assigned credential in a following SH protocol. Authenticity is provided on matched elements in this protocol since they are associated with the credential by using the homomorphism property in SH protocol. In the second interlocked protocol a SH protocol is executed first and credentials with same group secret are verified if the handshake succeeds. Then a common value is generated based on the verified credentials. This value is used in a following PSI protocol as common homomorphic random number. Detection resistance and impersonation resistance are provided in this protocol since adversaries cannot launched the detection and impersonation attack if they cannot compromise the executed SH protocol and know the common value.

B. A SECRET HANDSHAKE PROTOCOL WITH MATCHING ELEMENTS

In this section, we present our secret handshake protocol with matching elements. It means that we execute a private set intersection protocol first and compute the matching elements, and then we execute a secret handshake for privacy-preserving authentication of both credentials and matching elements. We use the credentials that have the matching elements associated with the group secret. First we present the parameters that are required in the protocol as $(q, G_1, G_2, e, H_1, H_2)$: q is a large prime number, G_1 denotes an additive cyclic group of prime order q, G_2 denotes a multiplicative cyclic group of the same order q, H_1 is a collision-free hash function that maps a string with arbitrary length to an element in G_1, H_2 is a collision-free hash function that maps a string with arbitrary length to a string with fixed length, and e denotes a bilinear map. G_1 and G_2 are selected in such a way that DLP is assumed to be hard in both of them.

Definition V.1 Homomorphic Randomization Function in SH Protocol with Matching Elements. Suppose user Alice joins a group g_A by obtaining a credential $g_A H_1(id_A)$, where $g_A \in G_2$ represents the group secret and $id_A \in G_1$ represents Alice's assigned pseudonym. Suppose Alice also has some matching elements with another user Bob as $(xy_1, xy_2, \ldots, xy_w)$. Alice selects a large random integer $r_{A1} \in G_2$, and uses the addition operation defined in G_1 to compute $r_{A1} \cdot H_2(xy_1 || xy_2 || \ldots || xy_w) \cdot g_A \cdot H_1(id_A)$.

As described in Fig. 5, there are 10 steps between Alice and Bob to execute the authentication protocol with matching elements. We illustrate the process on Alice's side since Bob's side is similar. After Alice withdrew her credential from the central authority, she executes the matching protocol with Bob and computes the intersection of their input sets. If the number of common elements is beyond a threshold, she continues interact with Bob by executing the authentication protocol. During the execution she associates the intersection with the group secret issued from the CA. Then after secret handshake if it succeeds, then Bob has the same group membership with and the intersection and Alice can continue any further interaction. Otherwise, either Bob does not own the same group membership or he is an adversary launching man-in-the-middle attack.

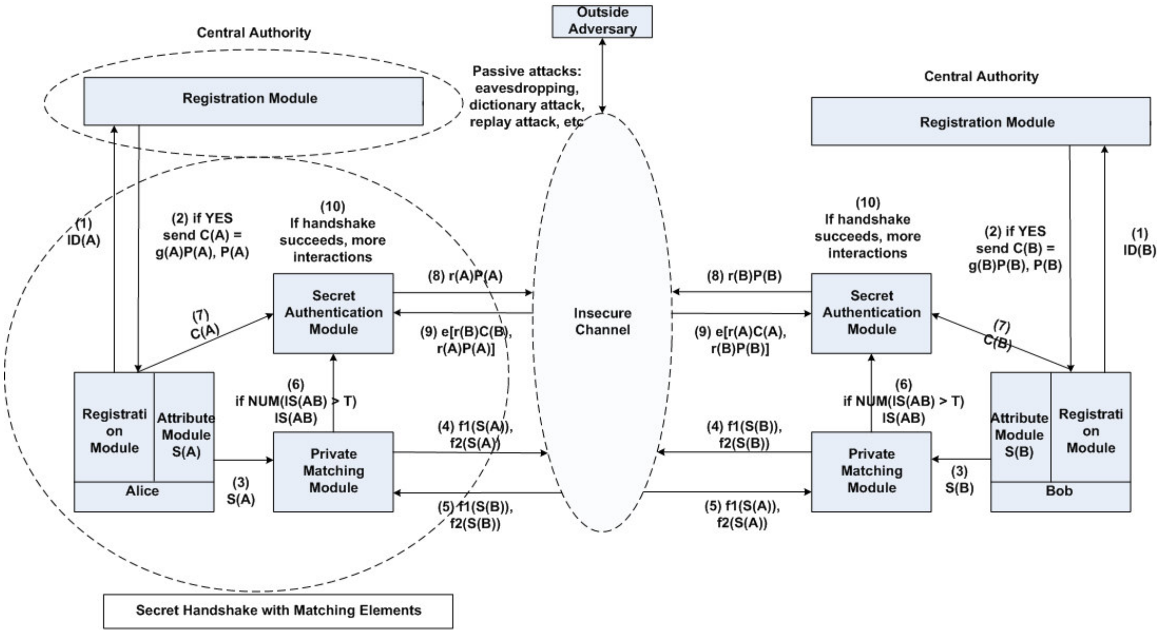

Central Authority
Registration Module
Outside Adversary
Passive attacks: eavesdropping, dictionary attack, replay attack, etc
Central Authority
Registration Module
(1) ID(A)
(2) if YES send C(A) = g(A)P(A), P(A)
(10) If handshake succeeds, more interactions
(10) If handshake succeeds, more interactions
(2) if YES send C(B) = g(B)P(B), P(B)
(1) ID(B)
(8) r(A)P(A)
(8) r(B)P(B)
Secret Authentication Module
Insecure Channel
Secret Authentication Module
(7) C(A)
(9) e[r(B)C(B), r(A)P(A)]
(9) e[r(A)C(A), r(B)P(B)]
(7) C(B)
(6) if NUM(IS(AB) > T) IS(AB)
(6) if NUM(IS(AB) > T) IS(AB)
Registration Module
Attribute Module S(A)
Alice
(3) S(A)
Private Matching Module
(4) f1(S(A)), f2(S(A))
(5) f1(S(B)), f2(S(B))
(4) f1(S(B)), f2(S(B))
(5) f1(S(A)), f2(S(A))
Private Matching Module
(3) S(B)
Attribute Module S(B)
Registration Module
Bob
Secret Handshake with Matching Elements

Fig. 5. A secret handshake protocol with matching elements.

Protocol IV. A Secret Handshake Protocol with Matching Elements

Our proposed protocol has three phases: *Protocol Initialization*, *Group Secret Mapping and Authentication Computing*. In the follows we present the detail of the three phases.

Phase I Protocol Initialization

The CA prepares the system parameters $(q, G_1, G_2, e, H_1, H_2)$ and a series of group secrets $[g_1, g_2, \ldots]$. The group secrets are known by the CA only. Other parameters are published to users. Alice requests the CA to join the group g_A. The CA verifies Alice's user identity to decide whether Alice can join the group. If yes, the CA grants the group membership to Alice by computing the credentials $cred_A = g_A \cdot H_1(id_A)$. Alice can use this credential to prove that it belongs to group g_A to other users in the same group. Suppose Alice and Bob have matching elements $(xy_1, xy_2, \ldots, xy_w)$. If w is larger than a threshold determined by Alice, she chooses one large random number $r_{A1} \in G_2$ and calls the homomorphic randomization function to compute $r_{A1}H_2(xy_1 || xy_2 || \ldots || xy_n)g_A H_1(id_A)$ directly. Similarly, Bob has his credential $cred_B = g_B \cdot H_1(id_B)$ and computes $r_{B1}H_2(xy_1 || xy_2 || \ldots || xy_n) \cdot g_B \cdot H_1(id_B)$. Alice and Bob randomly generate two non-zero integers, r_{A2} and r_{B2}, to prevent replay attack. Detailed interactions of our secret handshake protocol are described as follows.

Phase II Secret Mapping and Phase III Result Computing

(1) Alice →Bob: r_{A2}, $rd_A = r_{A1} \cdot H_1(id_A)$

(2) Bob: Compute $V_B = H_2(U_B || r_{A2} || r_{B2} || 0)$, $U_B = e(rd_A, r_{B1}H_2(xy_1 || xy_2 || \ldots || xy_w)g_B H_1(id_B))$. *Here Bob implements the secret mapping function to map his own credential* $cred_B = g_B \cdot H_1(id_B)$ *from an element in* G_1 *to an element in* G_2 *through bilinear map* e.

(3) Bob →Alice: r_{B2}, $rd_B = r_{B1}H_1(id_B)$, V_B

(4) Alice: $V_{B(A)} = H_2(U'_B || r_{A2} || r_{B2} || 0)$, $U'_B = e(rd_B, r_{A1}H_2(xy_1 || xy_2 || \ldots || xy_w)g_A H_1(id_A))$. *If* $V_B = V_{B(A)}$, *then Alice knows Bob belongs to the same group, i.e.,* $g_A = g_B$, *and verifies Bob's matching elements* $H_2(xy_1 || xy_2 || \ldots || xy_w)$. *Otherwise, Bob belongs to a different group, i.e.,* $g_A \neq g_B$ *or Bob does not own the matching elements. Here Alice implements the result computing function to find out whether* $H_2(U'_B || r_{A2} || r_{B2} || 0) = H_2(U_B || r_{A2} || r_{B2} || 0)$.

(5) Alice →Bob: $V_A = H_2(U'_B || r_{A2} || r_{B2} || 1)$

(6) *Bob: Compute* $V_{A(B)} = H_2(U_B \| r_{A2} \| r_{B2} \| 1)$. *If* $V_A = V_{A(B)}$, *Bob knows that Alice belongs to the same group, i.e.,* $g_A = g_B$, *and verifies Alice's matching elements* $H_2(xy_1 \| xy_2 \| \ldots \| xy_w)$. *Otherwise, Alice belongs to a different group, i.e.,* $g_A \neq g_B$ *or Alice does not own the matching elements.*

Theorem V.1. **Authenticity on Matching Elements in SH Protocol with Matching Elements**

A privacy-preserving authentication protocol that uses homomorphic randomization function in protocol initialization phase and associates matching elements with assigned credentials guarantees authenticity on the matching elements.

Proof: Suppose Alice and Bob execute a PSI protocol and obtain their matching element as $(xy_1, xy_2, \ldots, xy_w)$. Suppose there is an adversary Trudy, who executes a classic man-in-the-middle attack to Alice and Bob. Trudy intercepts the messages transmitted between Alice and Bob and acts as Alice to Bob and Bob to Alice. Since in the proposed PSI protocol Alice (Bob) modifies Bob's (Alice's) ciphertexts by mapping his (her) own elements into hidden values and associating the values with the ciphertexts, the transmitted messages contain both Alice's and Bob's mapped elements. As a result, Trudy can successfully prove to Alice (Bob) that he has the matching elements as Bob (Alice) does. In our new secret handshake protocol, Alice (Bob) computes verification value as $r_{A1}H_2(xy_1 \| xy_2 \| \ldots \| xy_w)g_AH_1(id_A)$ $(r_{B1}H_2(xy_1 \| xy_2 \| \ldots \| xy_n)g_BH_1(id_B))$ *by* associating the matching elements with assigned credentials. Since in our secret handshake protocol no sensitive information are contained in the transmitted values and r_{A2} and r_{B2} are included to generate single-use verification value, Trudy cannot successfully launch man-in-the-middle attack as long as he cannot deduce $H_2(xy_1||xy_2||\ldots||xy_n)$ from $r_{A1}H_2(xy_1||xy_2||\ldots||xy_w)g_AH_1(id_A)$ *or* $r_{B1}H_2(xy_1||xy_2||\ldots||xy_n)g_BH_1(id_B))$. As a result, Trudy cannot verify himself as the owner of $(xy_1, xy_2, \ldots, xy_w)$ and our protocol guarantees authenticity of matching elements on homomorphic randomization.

C. A PSI PROTOCOL WITH VERIFIED CREDENTIALS

In this section, we present our private set intersection protocol with verified credentials. It means that we execute a secret handshake protocol first and verify the assigned credentials, and then we execute a private set intersection for privacy-preserving matching of elements. First we present the parameters that are required in the protocol as $(q, G_1, G_2, e, H_1, H_2)$. Here q is a large prime number, G_1 denotes an additive cyclic group of prime order q, G_2 denotes a multiplicative cyclic group of the same order q, H_1 is a collision-free hash function that maps a string with arbitrary length to an element in G_1, H_2 is a collision-free hash function that maps a string with arbitrary length to a string with fixed length, and e denotes a bilinear map. G_1 and G_2 are selected in such a way that DLP is assumed to be hard in both of them.

The main idea to execute an authentication protocol following a matching protocol for Alice and Bob is to use the SH protocol to verify each other's group membership first and then utilize the verification outcome to protect the execution of the matching protocol in two ways. If the secret handshake fails, Alice (Bob) can terminate the interaction with Bob (Alice) since they do not belong to the same group. If the handshake succeeds, Alice and Bob can manipulate the common verification value to generate pairwise session keys to build a secure communication channel for further execution of matching protocols. In addition, they can associate the verification value with their matching process to defend against impersonation attack. In the follows we present the detail.

Definition V.2 Homomorphic Randomization Function in PSI Protocol with Verified Credentials. Suppose Alice joins group g_A by obtaining a credential $g_A H_1(id_A)$, where $g_A \in G_2$ represents the group secret and id_A represents Alice's assigned pseudonym. After Alice executed a SH protocol with Bob, they can compute some common verification V_{AB} if the handshake succeeds. Suppose Alice will also execute a PSI protocol with Bob. Alice associates the verification value $H_2(V_{AB}||0)$ as a large random number with her encrypted elements. She puts $H_2(V_{AB}||0)$ only in back points of her ciphertext.

Protocol V. A Private Set Intersection Protocol with Verified Credential

Our proposed protocol has three phases: *Protocol Initialization*, *Element Mapping and Matching Result Computing.* In the follows we present the detail of the three phases.

Phase I. Protocol Initialization

The CA prepares the system parameters $(q, G_1, G_2, e, H_1, H_2)$ and a series of group secrets $[g_1, g_2, \ldots]$. The group secrets are known by the CA only. Other parameters are published to users. Alice requests the CA to join the group g_A. The CA verifies Alice's user identity to decide whether Alice can join the group. If yes, the CA grants the group membership to Alice by computing the credentials $[g_A \cdot H_1(id_A)]$. Alice can use this credential to prove that it belongs to group g_A to other users in the same group. Similarly Bob also has his credential $[g_B \cdot H_1(id_B)]$. Alice and Bob execute a SH handshake protocol to verify whether $g_A = g_B$, i.e., whether they have the same group membership. If no, Alice (Bob) can choose to terminate the interaction with Bob (Alice). If yes, they can use the common verification value to generate a pairwise session key as $V_{AB} = H_2(U_A||r_{A2}||r_{B2}||2) = H_2(U_B||r_{A2}||r_{B2}||2)$. Then Alice and Bob can use a symmetric key cryptosystem like AES to build a secure channel between them with V_{AB} as the encryption and decryption key. That said, all the transmitted messages presented below are encrypted and decrypted using V_{AB} as the key.

Suppose Alice has element set $X = (x_1, x_2, \ldots, x_n)$ and Bob has element set $Y = (y_1, y_2, \ldots, y_m)$ where n and m are two chosen integers. In the follows we only describe Alice's matching procedure since Bob's is similar. First Alice generates her pairwise matching keys (k_{EA}, k_{DA}). k_{EA} is used for encryption and the latter one is used for decryption. Then Alice takes the input elements X and uses her k_{EA} to encrypt them to get the ciphertexts $E_A(X)$. The ciphertexts are then sent to Bob. After receiving $E_A(X)$, Bob modifies Alice's ciphertexts and gets $M_B(E_A(X) = M_B(Y, E_A(X))$ where Y is Bob's input elements and $M()$ denotes a modification function with both Y and $E_A(X)$ as inputs. Bob then sends the modified ciphertexts back to Alice. After receiving the modified encrypted elements, Alice uses the decryption function to decrypt the modified ciphertexts as $D_A(M_B(E_A(X)))$. Common elements between X and Y will be discovered after the decryption.

In the follows we present the detail of the first phase. First Alice generates her pair of matching keys (k_{EA}, k_{DA}) based on the base element $P_B \in G_1$: $k_{EA} = k_{DA} \times P_B$, where $k_{DA} \in Z_q$ and $k_{EA} \in G_1$. *Bob* also generates his key pair in a similar way, i.e., $k_{EB} = k_{DB} \times P_B$. Before exchange their elements for matching, Alice generates two large random integers r_{A1}, $r_{A2} \in Z_q$. Bob also generates his random integers r_{B1}, $r_{B2} \in Z_q$. In addition, Alice computes $H_2(V_{AB}||0)$ as another parameter for the matching process. First Alice needs to transfer her elements to large integers, i.e., $H_2(X) = [H_2(x_1), H_2(x_2)\ldots H_2(x_n)]$. She then uses the encryption key k_{EA} and $H_2(V_{AB}||0)$ to generate her ciphertexts as $E_A(X) = [E_A(x_1), E_A(x_2), \ldots, E_A(x_n)]$, where $\forall i \in [1, n]$, $E_A(x_i) = (E_{Af}, E_A(x_i)_b)= (r_{A1}P_B, r_{A1}H_2(a_i)H_2(V_{AB}||0)k_{EA})$. Bob also generates his ciphertexts in a similar way. He computes $H_2(V_{AB}||1)$ as his parameter in the matching process. He then generates his ciphertexts in a similar way $E_B(Y) = [E_B(y_1), E_B(y_2), \ldots, E_B(y_m)]$, where $\forall j \in [1, m]$, $E_B(y_j) = (E_{Bf}, E_B(y_j)_b)= (r_{B1}P_B, r_{B1}H_2(b_j)H_2(V_{AB}||1)k_{EB})$. The detail of the privacy-preserving elements matching protocol is presented as follows:

Phase II Secret Mapping and Phase III Result Computing

(1) Alice $\rightarrow$ *Bob:* $E_A(X)$

(2) Bob $\rightarrow$ *Alice:* $E_B(Y)$

(3) Alice: compute $H_2(V_{AB}||1)$ *and choose* $r_{A2} \in Z_q$*, then compute* $M_A(E_B(Y))$ *by using the homomorphic randomization function* $M_A(E_B(Y)) = [M_A(E_B(y_1)),\ldots, M_A(E_B(y_m))] = [M_A(E_{Bf})_1,\ldots, M_A(E_{Bf})_n)], [M_A(E_B(y_1)_b), \ldots, M_A(E_B(y_m)_b)]$, $\forall i \in [1, n]$, $M_A(E_{Bf})_i = r_{A2}H_2(a_i)H_2(V_{AB}||1)E_{Bf} = r_{A2}r_{B1}H_2(a_i)H_2(V_{AB}||1)P_B$; $\forall j \in [1, m]$, $M_A(E_B(y_j)_b) = r_{A2}E_B(y_j)_b= r_{A2}r_{B1}H_2(b_j)\, H_2(V_{AB}||1)k_{EB}$

(4) Bob: compute $H_2(V_{AB}||0)$ *and choose* $r_{B2} \in Z_q$, *then compute* $M_B(E_A(X))$ *by using the homomorphic randomization function* $M_B(E_A(X)) = [M_B(E_A(x_1)),\ldots, M_B(E_A(x_n))] = [M_B(E_{Af})_1,\ldots, M_B(E_{Af})_m)], [M_B(E_A(x_1)_b), \ldots, M_B(E_A(x_n)_b)], \forall j \in [1, m], M_B(E_{Af})_j = r_{B2}H_2(b_j)H_2(V_{AB}||0)E_{Af} = r_{B2}r_{A1}H_2(b_j)H_2(V_{AB}||0)P_B; \forall i \in [1, n], M_B(E_A(x_i)_b) = r_{B2}E_A(x_i)_b) = r_{B2}r_{A1}H_2(a_i) H_2(V_{AB}||0)k_{EA}$

(5) Alice $\rightarrow$ *Bob:* $M_A(E_B(Y))$

(6) Bob $\rightarrow$ *Alice:* $M_B(E_A(X))$

(7) Alice: for $j = 1$ *to* m, *compute* $k_{DA}M_B(E_{Af})_j$, *for* $i = 1$ *to* n, $j = 1$ *to* m, *test whether* $k_{DA}M_B(E_{Af})_j = M_B(E_A(x_i)_b)$. *If yes, put the corresponding element into* $X \cap Y$ *as the intersection set derived by Alice;*

(8) *Bob: for* $i = 1$ *to* n, *compute* $k_{DB}M_A(E_{Bf})_i$, *for* $i = 1$ *to* n, $j = 1$ *to* m, *test whether* $k_{DB}M_A(E_{Bf})_i = M_A(E_B(y_j)_b)$. *If yes, put the corresponding element into* $X \cap Y$ *as the intersection set derived by Bob;*

Theorem V.2. **Detection Resistance and Impersonation Resistance on Matching Elements in PSI Protocol with Verified Credential**

A privacy-preserving element matching protocol that uses homomorphic randomization function in protocol initialization phase and associates verified credential with matching elements guarantees impersonation attack resistance on the matching elements.

Proof: Suppose Alice and Bob execute a SH protocol, and the handshake succeeds. Alice and Bob can use the value to generate a secret pairwise session key $V_{AB} = H_2(U_A||r_{A2}||r_{B2}||2) = H_2(U_B||r_{A2}||r_{B2}||2)$, and use the key to build a secure communication channel by encrypting and decrypting all transmitted messages. The first part of Theorem V.2 is obvious since only Alice and Bob know the session key and no passive attack can be successfully launched on messages transmitted in the secure channel. Suppose there is an adversary Trudy, who executes an impersonation attack by launching attacks like replay attack to Alice and/or Bob. Trudy intercepts the messages transmitted between Alice and Bob. Trudy then acts as a legitimate party by sending Bob's initialization message to Alice or Alice's initialization message to Bob. Since in the original PSI protocol Alice (Bob) modifies Bob's (Alice's) ciphertexts by mapping his (her) own elements into hidden values and associating the values with the ciphertexts, the transmitted messages contain both Alice's and Bob's mapped elements. As a result, Trudy can successfully prove to Alice (Bob) that he has the matching elements. In our new protocol, Alice (Bob) computes ciphertexts of matching elements as $E_A(x_i)=(E_{Af},E_A(x_i)_b)=(r_{A1}P_B,r_{A1}H_2(a_i)H_2(V_{AB}||0)k_{EA})$ $(E_B(y_j)=(E_{Bf},E_B(y_j)_b)=(r_{B1}P_B,r_{B1}H_2(b_j)H_2(V_{AB}||1)k_{EB}))$ *by* putting the common verification value in the latter ciphertexts. Since in our secret handshake protocol no sensitive information are contained in the transmitted values and r_{A2} and r_{B2} are included to generate single-use verification value, Trudy cannot successfully launch replay attack as long as he cannot know the V_A or V_B or V_{AB}. As a result, our protocol guarantees detection resistance and impersonation attack resistance on matching elements.

VI. APPLICATIONS ON PRIVACY-PRESERVING PROTOCOLS

In Section VI the proposed PSI and SH protocols are implemented to build privacy-preserving inquiry matching (PPIM) for social network applications and privacy-preserving alert correlation (PAC) independent network alert sources.

A. SOCIAL PRIVACY-PRESERVING INQUIRY MATCHING APPLICATION

Social networking services (SNS), e.g., Facebook [29], Opensocial [47], Match.com [48], LinkedIn [49] etc., are rapidly expanding from their casual usage of making social contacts to more serious applications of building professional contacts, *e.g.,* PartnerUp [50], Sermo.com [51], INmobile.org [52], etc. The rich collection of personal and contact information in SNS systems makes them the prime target of adversary. As the industry gradually yet surely pushes SNS to the next level of high-value applications, it is critical to examine the privacy management issues in order to protect consumer privacy without sacrificing the convenience of existing systems.

A typical system architecture for most SNS is depicted in Fig. 6, where an SNS vendor makes its web portal available for free or paid services, i.e., posting and browsing of photos, trivia, personal, or business information. Consumers can use the *discovery/search* service in SNS to locate potential contacts using typical search cues, such as personal interests, names, or locations, etc., and the SNS providers may also make contact suggestions based on consumer profiles. Users can follow search outcomes to make new contacts based on a simple handshaking protocol, “friend invitation and approval”. A business related SNS usually has more rigorous control on user registration, e.g., Sermo.com [51] for physicians and INmobile.org [52] for senior professionals of the wireless industry. While this type of SNS vendors usually employ registration screening, pseudonyms and message encryption to protect user privacy, there is no broadly adopted privacy preserving matching techniques to protect the inquires being used in matching processes.

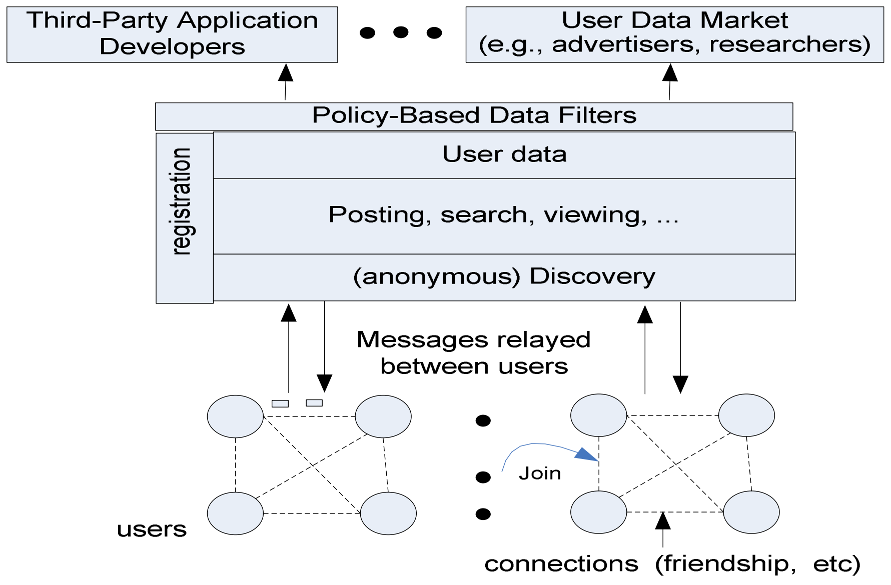

Third-Party Application Developers
User Data Market (e.g., advertisers, researchers)
Policy-Based Data Filters
registration
User data
Posting, search, viewing, ...
(anonymous) Discovery
Messages relayed between users
Join
users
connections (friendship, etc)

Fig. 6. A typical SNS architecture.

Privacy breaching can happen in many different ways, and it is hard to contain them completely. By agreeing to the service policies, consumers knowingly or unknowingly permit the SNS providers to make supposedly sanitized/aggregated data [53][54] available to third party application developers [55], or the general data market [54]. However, in practice, the adversary can still recover sensitive information from the sanitized data, e.g., professional aggregators [56][57][58], de-anonymizer tool [59] for employers [60], decoding of social connection based on P2P techniques [61]. More seriously, third party vendors may not have as high standards on consumer privacy protection as the leading vendors [62][63][64]. As an increasing number of privacy breaching incidents begin to emerge, either by exploiting system vulnerabilities [65][66][67][68], or accidental or intentional release of personal data [69][70][71], we claim that the overall security strength of existing SNS is not adequate to support current social networking applications. Many designs have been proposed, e.g., [2][5][40][62][72][73][74][75][76] to address those issues. Most of them are to allow user to control their own private information and deploy the security policy based on their own requirement, as illustrated in Fig. 7. The problem is that they are primarily focused on the last step, e.g., how to allow a user to control the content/profiles shared with his matched friends so that they do not leak unnecessary information [5][77]. Given the fact that SNS providers usually control the phases of user registration and user discovery, the privacy protection for the middle step is largely ignored, i.e., how to find matched and/or authenticated friends in a privacy-preserving manner. Without this, users may already leak their sensitive information before they can control their shared/public social data.

In this section we proposed a new system, *privacy-preserving inquiry matching* (PPIM), to address the problem. In PPIM, *attributes* of *matching inquiries* are considered highly privileged information to the owners, e.g., new drugs [51], product pricing [52], personal health/finance status [78], etc. Often, *group affiliation* is also considered privileged information [54]. As such, a principal may only allow peers to establish connection when he/she (1) has the same inquires attributes, (2) registered to the same group, or (3) both of above. Conceptually, one could treat matching of attributes and verification of membership affiliation problems as two independent problems, and they can be solved using the existing *private set intersection* (PSI) protocol [24] and the *secret handshaking* (SH) algorithm [25], respectively. Or, a user can verify the membership with others first before proceeding to matching of privileged attributes. However, the PSI algorithm alone is vulnerable to man-in-the-middle attack [79] and replay attack [80], because the adversary can first passively relay packets between two users until they made successful making, and then become active to impersonate either/both of them to hijack subsequent communications. Moreover, the original SH algorithm may not be scalable for large systems because a user needs to request and store a large number of single-use pseudonyms to maintain anonymity and unlinkability, which is crucial for protecting group affiliation information.

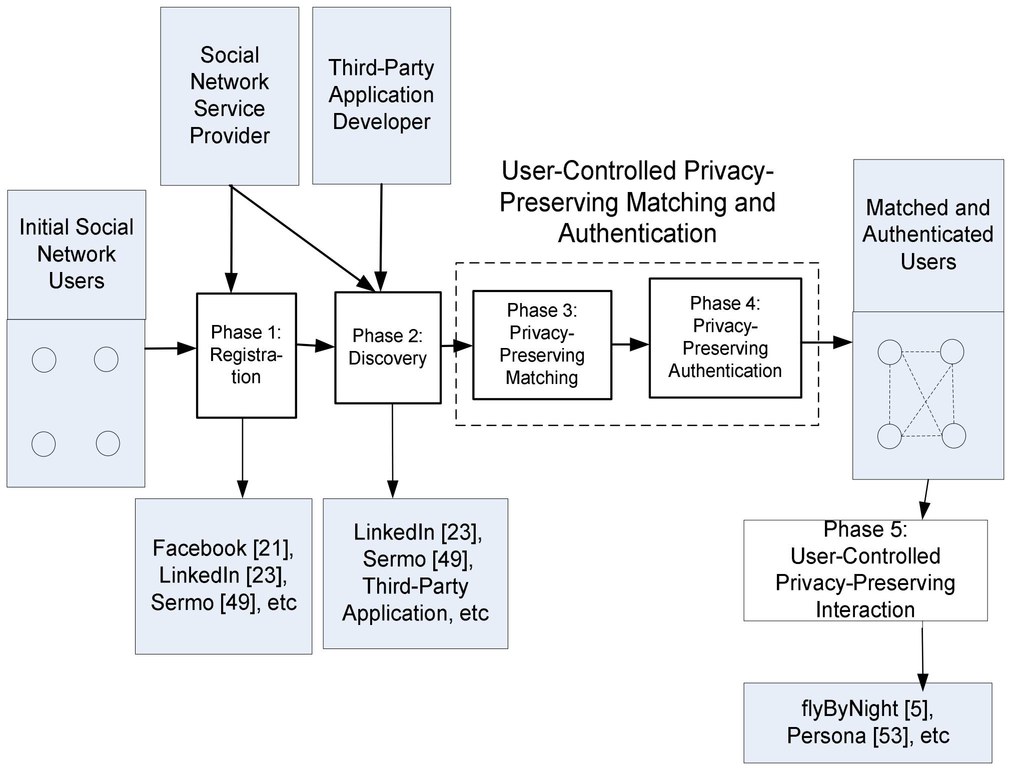

Social Network Service Provider
Third-Party Application Developer
User-Controlled Privacy-Preserving Matching and Authentication
Initial Social Network Users
Matched and Authenticated Users
Phase 1: Registra-tion
Phase 2: Discovery
Phase 3: Privacy-Preserving Matching
Phase 4: Privacy-Preserving Authentication
Facebook [21], LinkedIn [23], Sermo [49], etc
LinkedIn [23], Sermo [49], Third-Party Application, etc
Phase 5: User-Controlled Privacy-Preserving Interaction
flyByNight [5], Persona [53], etc

Fig. 7. Current privacy protections for SNS.

We propose to use the proposed interlocked protocol to enhance PSI and SH algorithms so that they can more effectively serve the needs of PPIM applications. There are three steps to use the interlocked protocol on PPIM. The first step is to implement the PSI algorithm using the elliptic curve cryptography (*ECC*) cryptosystem, in order to lower its computing costs. The second step is to use the proposed unlinkable secret handshake protocol so that consumers just need to register once from a particular *Group Administrator* (GA) to engage in PPIM activities. The third step is to interlock the PSI and SH by proper selection of the elliptic curve parameters and functions, and making the outputs produced in PSI as a part of the computing inputs to SH. Two principals engaged in a PPIM transaction could first run the PSI protocol, and then feed the matched results (in a derived form) to the SH protocol for authentication. Or, they could first run SH protocol to authenticate each other, and then use a common “session key” derived from the SH to make it a part of the attributes, so that the adversary cannot engage in the PSI protocol without the common key. By bundling SH with PSI, the proposed PPIM protocol can defend against the man-in-the-middle attack [79] and replay attack [80]. Running PSI first and then SH, vs., SH first and then PSI some subtle differences in their security implications, but both are equally doable. We will focus on the PSI-SH, but the theoretical basis is also applicable to the SH-PSI design. Another advantage of our design is that two users can generate a pairwise session key after a successful PSI-SH execution. This way, the PPIM system can directly cooperate with systems like flyByNight [5] or Persona [77] without the need of a complex key distribution infrastructure.

A common concern for any on-line applications is the notion of Sybil attack [66][81], in which an adversary uses ghost accounts to gain unduly benefits, e.g., starting a hype, gaining control of a virtual group, etc. Many techniques have been proposed in the literature to detect such misbehaviors [82][83][84]. We assume that trade groups for high-value applications have stringent registration processes similar to those in [51][52] to eliminate ghost accounts in PPIM user community. The rigorous matching and authentication procedures make it much easier for GA to sanction against their misbehaved group members based on reputation or similar punitive systems.

The architecture of the proposed PPIM scheme and its relationship with respect to existing SNS systems is depicted in Fig. 8. The only additional infrastructure requirement for the PPIM service is a PPIM relay server, which can be a part of existing SNS, or an independent system. It serves the simple purpose of listing potential matching candidates, and relaying of PPIM related messages between two matching candidates. In plaintext, a PPIM inquiry Q consists of a set of *attributes* T and *credentials* C, $Q=(T,C)$, where $T=[t_0, t_1, \dots, t_n]$, and $C=[c_0, c_1, \dots, c_m]$, where c_i is issued by GA_i. The contents of T and C, including their sizes, are determined based on personal choices or some best practice of the trade. An element t_i in T represents one of the n items to be matched with others, and the user can choose any subset of T for a PPIM transaction. Each element c_i in C represents a point on a chosen elliptic curve generated from a number of expressions. Some derivatives of selected elements in C will be sent to other users during different PPIM transactions. Each user also has a unique identity, *ID*, in a group, from each registered GA. The ID will be solely used as a piece of private data to run the SH protocol, and it should not be confused with common user accounts.

In PPIM we propose to adopt existing SNS as the platform for strangers to make their first contacts, i.e., *discovery* of candidates using non-privileged information [29][47][49], peer recommendation [52], or pseudonyms [54], etc. From the list of candidates produced from the discovery process, users can proceed to the next level of (privacy-preserving) matching based on our proposed scheme. The PPIM relay server is interfaced to SNS, so that when users wish to proceed with PPIM matching, their public profiles can be posted. Once two users agree to enter a PPIM transaction to match their inquiries, all messages exchanged between them are essentially ciphertexts. We use the interactions between three users *Alice*, *Bob*, and *Mallory* to illustrate the overall system concept. Conceptually, any two users Alice and Bob exchange encrypted attributes through the PPIM relay server for the matching of $T^a = \{t_0^a, t_1^a, ..., t_n^a\}$ and $T^b = \{t_0^b, t_1^b, ..., t_{n'}^b\}$, and then verify if they have made the same PSI matching results, and also have one or more common affiliations (governed by the same GA). At end of a transaction, Alice and Bob can either affirmatively identify their matched attributes, with verified common membership; or neither of them will gain any hint about the attributes nor the membership of the counterpart. The transaction is immune from passive man-in-the-middle attack.

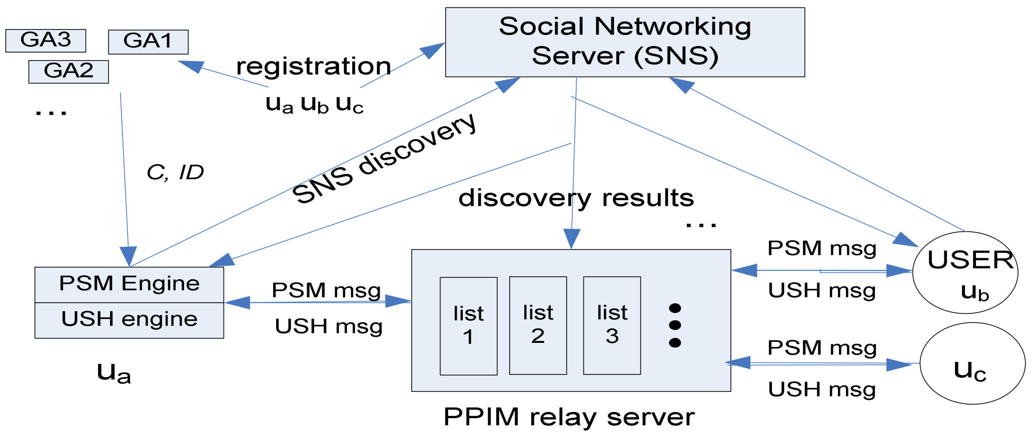


Fig. 8. Architecture of the PPIM service.

Overall, when Alice is engaged in a PPIM transaction, it needs to submit three types of messages (in addition to some other messages), $[E_a(M_a(T^a)); w_a(c_i^a); v_a(c_i^a)]$ for

PSI and SH protocols. $E_a(M_a(T^a))$ is the ciphertext of $M(T^a)$ generated by Alice, where *$M(T^a)$* is the results of mapping of T^a based on a *mapping function* ***M***.

Alice, Bob and Mallory exchange all PSI related messages through the PPIM server. We will use Alice to carry the rest of discussions, because identical arguments, except for the variables involved, are applicable to all users. Assuming that Alice first sends out $E_a(M_a(T^a))$ to Bob and Mallory. Then, Bob sends its own $E_b(M_b(T^b))$ and returns the *modified* inquiry $F_b(E_a(M_a(T^a)))$ back to Alice. $F_b(E_a(M_a(T^a)))$ is computed by Bob using a known homomorphic encryption method with T^b as its input. Similarly, Mallory also sends $E_m(M_m(T^m))$ and $F_m(E_m(M_m(T^a)))$ to Alice. Then, Alice can use the received messages to perform its PSI computing locally. Bob and Mallory can perform similar PSI computing after they receive their modified inquires.

For Alice to perform SH protocol with both Bob and Mallory, the related message exchanges are derived from the PSI matching outcomes, and the credentials of Alice. To verify that Alice and Bob are in the same group, Alice needs to prove that $v_a(c_i^a) = v_b^a(c_j^b)$, which denote verification values generated by Alice and Bob, respectively. $v_a(c_i^a)$ is generated by a *generation function* **G** based on bilinear pairing [45][85]. $v_b^a(c_j^b)$ is computed by the *verification function* **V** of Bob, based on $w_a(c_i^a)$ and Bob's own credential c_j^b. $w_a(c_i^a)$ (generated by Alice) is 2-tuple: (1) a single-use number n_a, and (2) the hashed, and then randomized ID_i^a ID_i^a, which is issued by GA$_i$ as Alice's ID in group$_i$.

Let $C^a = \{c_0^a, c_1^a, ..., c_m^a\}$ and $C^b = \{c_0^b, c_1^b, ..., c_{m'}^b\}$for denote credentials for Alice and Bob, respectively. We summarize the five security properties (P1 to P5) of PPIM scheme based on the scenario that Alice does PPIM transaction with Bob. For (Alice, Bob), all others are considered outsiders.

Attribute Confidentiality: Alice and Bob can find whether $T^a \cap T^b \geq \delta_a$, δ_b where δ_a, δ_b are threshold values chosen by Alice and Bob, respectively. *For all attributes* $t_i^a \in T^a$ and $t_i^a \notin T^a \cap T^b$, t_i^a is not revealed to Bob. Similarly, for all $t_j^b \notin T^a \cap T^b$, t_j^b is not revealed to Alice. T^a and T^b is not revealed to outsiders.

Attribute Authenticity: Alice and Bob can find whether or not $C^a \cap C^b \geq \delta_{a'}$, $\delta_{b'}$ where $\delta_{a'}$, $\delta_{b'}$ are threshold values chosen by Alice and Bob, respectively, where "$c_i^a = c_j^b$" means that they are issued by the same GA. For all credentials $c_i^a \in C^a$ and $c_i^a \notin C^a \cap C^b$, c_i^a is not revealed to Bob. Similarly, for all credentials $c_j^b \notin C^a \cap C^b$, c_j^b is not revealed to Alice. C^a and C^b is not revealed to outsiders.

Message Privacy: No message exchanged between Alice and Bob can be deciphered by outsiders.

Relationship Privacy: Outsiders cannot know whether or not Alice and Bob have a positive or negative match.

Unlinkability: No message can be linked to ID_i^a , ID_j^b .

The interlocked PSI protocol is the Protocol I proposed in Section III.B. To make it more efficient in PPIM, we made the following changes: first off, a supersingular elliptic curve [85] ς is used for both PSI and SH to reduce the complexity of curve selection. In actual implementation of *M*, we modified the original hash function in [86] with decreased parameter size so that attributes chosen from a relatively small space can be fit into the relatively smaller number of points on ς. We still adopted the classic hash function SHA-1 [87] as H_2. In addition, our system architecture is an end-user controlled privacy management model that does not require providing anyone privileged inquiry information, except those who can successfully complete PPIM transactions with the information owner.

The SH is the Protocol III proposed in Section IV. B. The SH is built upon the *bilinear Diffie-Hellman* (BDH) *intractability* assumption. By choosing the *Tate pairing* for SH design, it satisfies the *bilinearity* and *degeneracy* of BDH. Selection of the paring-based functions and parameters follows the discussions in [45].

Fig. 9 presents the flowchart of our ECC-based PSI scheme. When Alice and Bob exchange messages, they need to generate their public keys (PUB_a and PUB_b) and private keys (PRI_a and PRI_b) from the base element $Q \in G_1$, where G_1 denotes an additive cyclic group of prime order q (which is a large prime number). Here we choose the group of points on a pairing-friendly curve ς to be G_1, in which DLP is known as ECDLP [45]. It is required that $PUB_a = PRI_a \times Q$, where $PRI_a \in F_q$ and $PUB_a \in G_1$ (F denotes a finite filed). Similarly, $PUB_b = PRI_b \times Q$. PRI_a is chosen by Alice, and PRI_b by Bob. After Alice picks T^a, she uses a global *mapping function M* to generate $M_a(T^a) = \{s_0^a, s_1^a, ..., s_{n+1}^a\}$, which is encrypted to $E_a(M_a(T^a)) = \{E_a(s_0^a), E_a(s_1^a), ..., E_a(s_{n+1}^a)\}$ using PUB_a and an *encryption function* ***E.*** Alice sends $E_a(M_a(T^a))$ to Bob. After receiving it, Bob uses a *modification function F* to *modify* $E_a(M_a(T^a))$ into $F_b(E_a(M_a(T^a))) = \{F_b^a(t_0^b), F_b^a(t_1^b), ..., F_b^a(t_{n'}^b)\}$ with $E_a(M_a(T^a))$ and some of his own attribute data in T^b as the inputs to ***F*.** Here, ***F*** needs to satisfy the homomorphic encryption property. Next, Bob sends $F_b(E_a(M_a(T^a)))$ back to Alice, who can then decipher the received $F_b(E_a(M_a(T^a)))$ by using a *decryption function D* and her private key PRI_a. The deciphered outcomes represent identical hashed values of original attributes, and they can be readily used to produce $IS(T^a, T^b)$, the intersection of T^a and T^b.

Then, Alice chooses δ_a as a threshold and checks whether or not the number of elements in $IS(T^a, T^b)$, $NUM(IS(T^a, T^b))$, is larger than δ_a. If yes, $IS(T^a, T^b)$ is used as the input to the SH engine for credential authentication. Bob goes through a fairly similar process to match attributes submitted by Alice. The protocol satisfies the security properties P1, P3 and P4. Here we can only briefly highlight them as follows. For P1, Bob cannot deduce T^a because of the hardness of ECDLP in G_1, i.e. Bob cannot deduce r_a from r_aPUB_a (in step 2) and thus cannot get s_i^a from $s_i^a + r_aPUB_a$. Similarly, Alice cannot deduce T^b. Alice cannot deduce extra elements not in $IS(T^a, T^b)$ because those elements are hidden by the multiplication of r_b to $\sum_{i=0}^{n+1} H_2(t_j^b)^i (E_{il})$ in F. Bob also cannot deduce any extra elements not in SI_B for similar reasons. For P3, all messages transmitted are in encrypted format. Due to ECDLP is hard in G_1, message privacy is maintained. For P4, thresholds δ_a and δ_b are chosen by Alice and Bob respectively and whether the matching results satisfies each user's requirement, e.g., $NUM(IS(T^a, T^b)) \geq \delta_a$, is only known to them. The detail of the interlocked PSI protocol was presented in Section III. B and is omitted here.

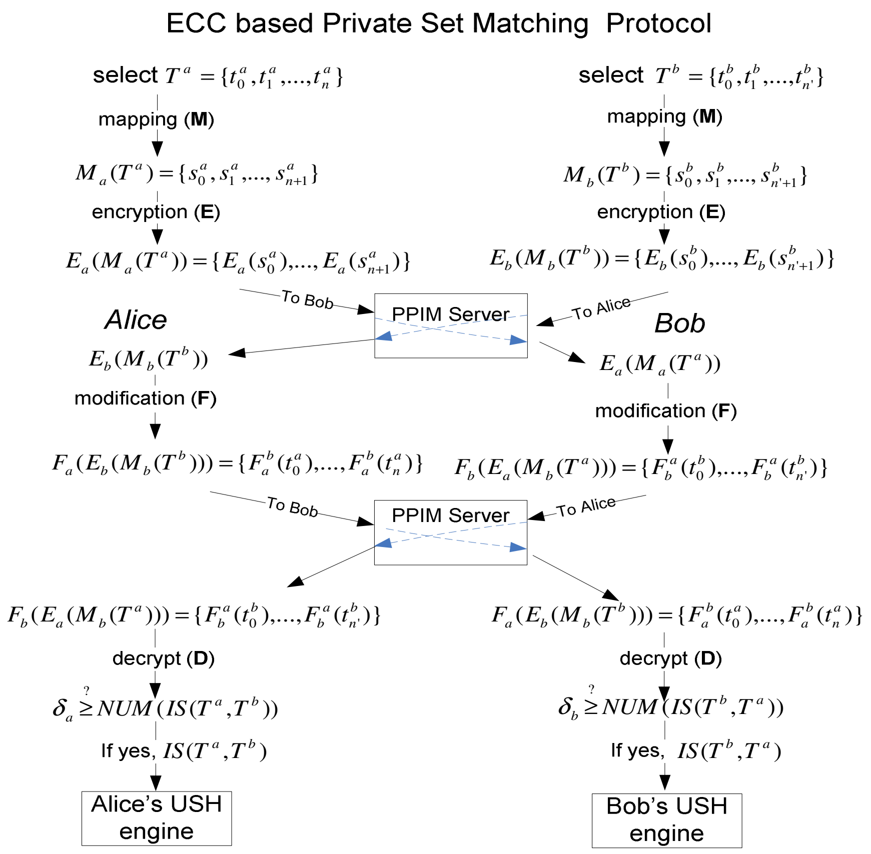

ECC based Private Set Matching Protocol
select $T^a = \{t_0^a, t_1^a, ..., t_n^a\}$
mapping (**M**)
$M_a(T^a) = \{s_0^a, s_1^a, ..., s_{n+1}^a\}$
encryption (**E**)
$E_a(M_a(T^a)) = \{E_a(s_0^a), ..., E_a(s_{n+1}^a)\}$
select $T^b = \{t_0^b, t_1^b, ..., t_{n'}^b\}$
mapping (**M**)
$M_b(T^b) = \{s_0^b, s_1^b, ..., s_{n'+1}^b\}$
encryption (**E**)
$E_b(M_b(T^b)) = \{E_b(s_0^b), ..., E_b(s_{n'+1}^b)\}$
To Bob
PPIM Server
To Alice
Alice
Bob
$E_b(M_b(T^b))$
modification (**F**)
$F_a(E_b(M_b(T^b))) = \{F_a^b(t_0^a), ..., F_a^b(t_n^a)\}$
$E_a(M_a(T^a))$
modification (**F**)
$F_b(E_a(M_b(T^a))) = \{F_b^a(t_0^b), ..., F_b^a(t_{n'}^b)\}$
To Bob
PPIM Server
To Alice
$F_b(E_a(M_b(T^a))) = \{F_b^a(t_0^b), ..., F_b^a(t_{n'}^b)\}$
decrypt (**D**)
$\delta_a \overset{?}{\geq} NUM(IS(T^a, T^b))$
If yes, $IS(T^a, T^b)$
Alice's USH engine
$F_a(E_b(M_b(T^b))) = \{F_a^b(t_0^a), ..., F_a^b(t_n^a)\}$
decrypt (**D**)
$\delta_b \overset{?}{\geq} NUM(IS(T^b, T^a))$
If yes, $IS(T^b, T^a)$
Bob's USH engine

Fig. 9. ECC-based PSI protocol in PPIM.

The SH protocol is illustrated in Fig. 10. A matched pair of Alice and Bob both have a common attribute set contained in *IS(T^a, T^b)* and *IS (T^b, T^a)*, but neither of them can tell if the PSI transaction was done with Mallory, i.e., the "man in the middle" adversary. If Mallory is actively engaged with other parties to complete PSI transactions without subsequent authentication, she could focus on collecting attributes as a form of intelligence gathering using numerous bogus accounts.

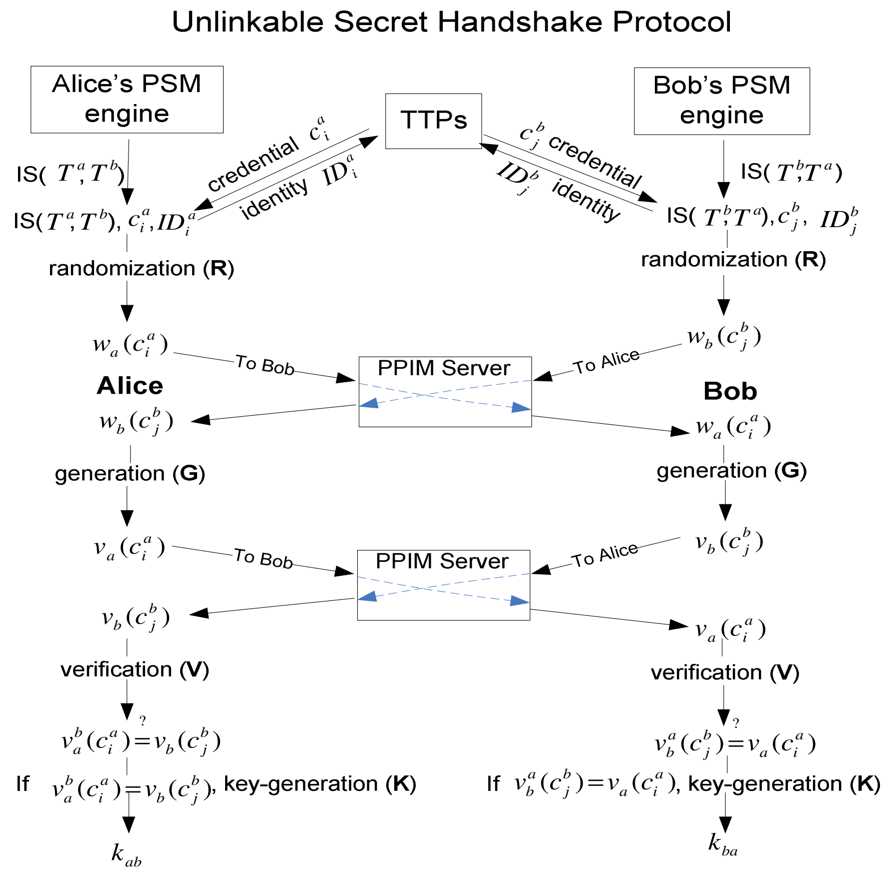

Unlinkable Secret Handshake Protocol
Alice's PSM engine
TTPs
Bob's PSM engine
IS(T^a, T^b)
credential c_i^a
identity ID_i^a
c_j^b credential
ID_j^b identity
IS(T^b, T^a)
IS(T^a, T^b), c_i^a, ID_i^a
IS(T^b, T^a), c_j^b, ID_j^b
randomization (R)
$w_a(c_i^a)$
To Bob
PPIM Server
To Alice
$w_b(c_j^b)$
Alice
Bob
generation (G)
$v_a(c_i^a)$
$v_b(c_j^b)$
verification (V)
$v_a^b(c_i^a) \overset{?}{=} v_b(c_j^b)$
$v_b^a(c_j^b) \overset{?}{=} v_a(c_i^a)$
If $v_a^b(c_i^a) = v_b(c_j^b)$, key-generation (K)
If $v_b^a(c_j^b) = v_a(c_i^a)$, key-generation (K)
k_{ab}
k_{ba}

Fig. 10. ECC-Based SH protocol in PPIM.

If Mallory is only eavesdropping, she cannot decipher the attributes, and therefore does not know *IS*(T^a, T^b), nor *IS* (T^b, T^a), but she could become active at end of the PSI protocol and to impersonate both Alice/Bob for subsequent communications. The SH is designed to eliminate such a possibility by using *IS*(T^a, T^b) (for Alice) and *IS* (T^b, T^a) (for Bob) as inputs to the SH protocol to prove that (1) they both are affiliated with the same group, and (2) they have the same intersection. Furthermore, at completion of the SH step, they can derive a common value which can then be used to produce a session key for their subsequent private communications. If group membership is not a critical concern, the SH can be used as a "confirmation protocol", where the PSI outcome is directly used as the "credential issued by GA" to prove the equality of $IS(T^a, T^b)$ and $IS(T^b, T^a)$ affirmatively. Neither Bob nor Alice will unveil $IS(T^a, T^b)$ or $IS(T^b, T^a)$ to each other if the SH produces a negative outcome. The detailed of the interlocked SH protocol was presented in Section IV. B and is omitted here.

Computing cost is an important issue which directly affects consumers' acceptance of new technologies. We will first briefly summarize the computing complexity of the propose scheme, and then will present experimental results of our prototypes. Knowing that they have much higher computing costs than other operations [86], we use the number of the point multiplications $T(P)$ and mappings $T(M)$ to estimate the performance of PPIM. For two users both with n attributes, it takes $(n + 1)$ mapping operations to transform n attributes to their corresponding elliptic curve points. It takes $(n + 2)$ point multiplications to encrypt them. It takes $[(n + 1)^2 + 1]$ point multiplications and $(n + 1)$ mappings to modify n ciphertexts. It needs $(n + 1)$ point multiplications to decrypt the modified ciphertexts to get the matching outcomes. The total cost of the PSI protocol is $O(PSI) \approx 2(n + 1)T(M) + [(n + 1)^2 + 2n + 4]T(P) \approx O(n)T(M) + O(n^2)T(P)$. The only major computing cost of SH is computing of Tate paring, and our secret handshake for the authentication of one credential needs 2 computations of pairing. The communication cost, or the total message size is proportional to number of the input attributes, or $O(n)$.

We consider the following attacks in the open networking environment: (1) active attacks that aim at compromising the authenticity of a user's private information, e.g. impersonation; (2) passive attacks that aim at compromising confidentiality of a user's private information, e.g. eavesdropping. We also consider two kinds of adversaries: outsider adversaries and inside adversaries. An outsider adversary does not know any of the roles in matching, nor does it have any knowledge about the user's private information. An inside adversary is a malicious user that participates in the PSI and SH protocols with some other user. We claim that our PPIM holds the security properties (P1 to P5) given the adversary model and attack model.

The PPIM solution focuses on preserving privacy for social matching process. That is, it is not intended to solve every security problems. Some attacks to our proposed system are possible. Here, we discuss some of these possible attacks and corresponding solutions. (1) Denial-of-Service attack: in this attack, a malicious user may launch arbitrary huge number of matching requests to other users, so that other users' computation and communication resources may be misused, like the effect of any DoS attack. A practical solution to this attack is to set a limit on the number of matching request from a single user to another one in a time window. (2) Absolute-Matching Attack: an attacker *A* may input all possible values of attributes into another user *B*'s requesting ciphertexts such that user *B* will always find matching attributes and consider *A* as a matching partner. One of the simple solutions is that our PPIM imposes a limit on the number of modified ciphertexts returned by *A* according to *B*'s number of encrypted attributes, e.g., the number of encrypted attributes and the number of returned modified ciphertexts must be same or their difference is within a small range.

To demonstrate the utility of the PPIM service, we developed a patient support community called “PatientMatch”, using Facebook as the SNS. The objective of the PatientMatch prototype is to form a supporting group among qualified constituents (patients, social workers, etc), for a particular type of health conditions (chronic illness, substance abuse, etc) to simulate the type of services provided in [78], but keep other people out of the group. Detail of the evaluation of PPIM is presented in Appendix B.

B. PRIVACY-PRESERVING CORRELATION OF NETWORK SECURITY ALERTS

Alert correlation refers to the process of identifying the true nature of an attack by discovering common (matched) attacking information from alerts generated from security tools deployed at different networks. Most basic alert attributes, e.g., IP addresses, packet payloads (which may contain user accounts, pin numbers), etc. are considered sensitive information [8][21]. Unrestricted sharing of (local) alerts may lead to malicious mining of network vulnerabilities [6][8][88], litigation, or even stealing of business secrets. In the mean time, matched alerts should be unveiled to involved parties to determine global attack information, but other unmatched local information should be protected. With the rapid growth in scales, speed, and sophistication of network attacks, there is a pressing need to develop useable tools for collaborative network administrators to correlate their local alerts to discover common attacks while protecting their privacy, i.e., privacy-preserving alert correlation schemes.

The majority of previous privacy-preserving alert correlation schemes [6][12][13][14]perform alert correlation using sanitized (e.g., hash, content generalization, etc) less sensitive data (e.g., source IPs), after sensitive alert data are removed. This approach often misses out the most critical information for recovery of developing situations yet is still vulnerable to dictionary attack [7], which is considered an important privacy threat for alert correlation [7][22]. Some alert correlation services are available for public access [89][90]. In these services, a central server collects alerts and publishes correlation results. Again, for privacy protection, users need to first remove sensitive information from their alerts before submitting them to the server. Moreover, in this architecture the adversary can use the server to launch probe-response attack and mapping attack [6][7], in order to identify the defense ability/vulnerability of targeted networks.

Private set intersection (PSI) (e.g., [24][32]), and secure multi-party computation (SMC) (e.g.,[41][42]) are two major types of crypto algorithms that allow users to use any sensitive data for privacy-preserving alert correlation. Through these algorithms users can know either (1) they have the same contents, or (2) they have unmatched contents without gaining any other information from other users. Numerous algorithms have been proposed for PSI and SMC, yet the cost of most of them is still considered too high (e.g., O(n2) crypto operations for PSI [24][32]) to be practical for large scale alert correlation applications [23][91].

How to guarantee privacy protection and make accurate alert correlation at low computing cost remains a major open problem [7][22]. As such, the design criteria of an alert correlation tool should have (1) guaranteed protection of privileged data, (2) accurate reflection of attack states from as many types of alert attributes as possible, (3) acceptable computing costs, (4)compatible with the distributed administration architecture of the Internet, and (5) ease of creating new applications.

We propose two cryptographic protocols for a privacy-preserving alert correlation (PAC) system to meet these five challenges. The first proposed is the linear PSI protocol (Protocol II) proposed in Section III. C. It is designed for privacy-preserving computing of the intersection of two sets of records for *formatted attributes*, i.e., attributes (IP addresses, port numbers, etc.) in unified formats. We will call the records that belong to a particular attribute "*attribute records*". For instance, Alice may want to inquire Bob "Which of these (source) IPs in my alerts also appear in your alerts?" Through Protocol II, both Alice and Bob can confirm the set of common IPs, and no other information related to the unmatched ones can be inferred from any of the exchanged messages. Existing PSI schemes would either use polynomial evaluation [24][32], or the bit-by-bit oblivious transfer (OT) [27] to find the intersection. On the other hand, Protocol II can perform matching through comparison of encrypted attribute records. A distinct feature of Protocol II is its linear computing cost for crypto operations (homomorphic encryption/decryption) with respect to the number of input records.

As the first of its kind, we also propose a new protocol (Protocol VI) that supports privacy-preserving computing of the *longest common sub-string* (LCS) of two records of a particular *string attribute* (e.g., captured payloads). The length for any record of a string attribute is bounded but arbitrary. Computing of the LCS of multiple suspected payloads can generate common contents (e.g., matched detection signatures of malwares) while remove local network information (unmatched contents). In Protocol VI, a variable-length string is first split into a set of concatenated, fixed length blocks. The LCS of the two strings is privately computed using the combination of Protocol II and location protection functions (crypto randomization and ciphertexts shuffling) on these blocks.

Both schemes are immune from passive attacks (e.g., dictionary attack) due to the well known *elliptic curve discrete logarithm problem* (ECDLP) [45] (design criterion 1). Accurate correlation results can be directly obtained for many types of attributes (design criterion 2). The proposed protocols have linear crypto computing complexity O(*n*). It takes 1 second (2 minutes) for our scheme to match between two sets of 100 (10k) records on a 2GHz Intel Xeon-based computer (design criterion 3). In comparison, it took required 213 seconds [23][26] for the original PSI algorithm [24], whose crypto computing complexity is ($O(n^2)$), to match between two sets of 100 records on a 3 GHz Intel Xeon-based computer. To meet the design criteria 4 and 5, at the network level PAC is based on a loosely coupled distributed architecture consisting of a *PAC communicator* and multiple *PAC clients.* Being a pair-wise matching architecture, PAC does not suffer from probe-response attacks and mapping attacks of the centralized correlation architecture.

PAC is the first of its kind that directly obtains true state information of attacks from the alert correlation process with guaranteed privacy protection against semi-honest adversary. It has among the lowest computing costs for this level of privacy protection. No published privacy-preserving alert correlation solution has achieved the same level of correlation accuracy, privacy protection, and computing costs simultaneously.

Protocol II follows the homomorphism based private set matching model initially proposed in [24], but it has one order lower of computing complexity, i.e., O(n), for crypto operations, than the polynomial evaluation technique in [24], i.e., O(n^2). Our Protocol VI is the first privacy-preserving LCS computing scheme with guaranteed privacy protection against semi-honest adversary.

PAC is designed for two well-behaved users (e.g., network administrators) to find (1) the intersection of two sets of attribute records, (2) the LCS of two records of any string attribute. Privacy preservation is guaranteed in PAC, i.e., all unmatched elements remain secret. All the records used in correlation process are assumed to be genuine, and verification of the authenticity of alert data is beyond the scope of this research. PAC is designed to resist passive attacks launched by honest-but-curious (i.e., semi-honest) adversary, but it is not designed to defend against active attacks, e.g., forging of alert records, traffic interception, etc.

Matching of alert records across networks can produce useful wide-area attack information. For instance, correlation of IP addresses can help discovering attack topologies. Matching of source IPs in alerts (on different networks) can help discovery of a common attacker. In this case, networks generating these alerts may be targets of a same attack. On the other hand, we can detect a common attack victim by matching of the destination IP addresses originated from different networks. Alert correlation can also help detection of *attack chains* (e.g., stepping-stone attacks), which refer to coordinated use of multiple hosts across different networks to hide trails of attacks. Traditionally, it often requires extensive coordination of network administrators across networks to manually filter and exchange data to capture the stealthy traffic. On the other hand, with the help of PAC network administrators can match source and destination IPs of detection alerts to discover the adjacent nodes of an attack chain, without the concern of unveiling any other unrelated (local) information. Fragment information can then be aggregated to reveal the global topology of the attack chain.

LCS based matching can be used for discovery of common content of raw packets, worms/bots signatures already generated by tools (e.g., Autograph [92]) within local networks, etc. Computation of the LCS from signatures across two networks gives more accurate, more global worms/bots signature by elimination of substrings likely related to local information.

The architecture of a PAC system is illustrated in Fig. 11(a). The PAC communicator serves as a community portal for participants to identify correlation partners using common tools such as instant messenger, email, posting, etc. Once two users agree to correlate their alerts, they can use the communicator to first settle the syntax, types, formats, and other information for the alert contents to be correlated. Then, all private computing functions are performed on client tools that run on the host machines of users.

The PAC client is consisted of three major modules: an *alert parser*, a *privacy-preserving matching engine* (PPME) and an *alert correlator* (AC), see Fig. 11(b). The alert parser extracts and format alert records from raw alerts generated from any local alerting tools. The PPME takes the attribute records as inputs and interact with other PPMEs to match them. Let $L_A(L_B)$ denote alerts owned by Alice (Bob), generated from her(his) local alerting systems. They are fed to the alert parser to extract a set of attribute records $A = (a_1, a_2, \ldots, a_n)$ $(B = (b_1, b_2, \ldots, b_m))$. Let $C=\{c_1, c_2, \ldots c_k\}$ denote the intersection of A and B. The objective of Protocol II in PPME is for Alice and Bob to identify C, but nothing else, provided that A and B are in the same format. Protocol VI in PPME is for Alice and Bob to privately compute the LCS of their input strings S_A and S_B, without unveiling any other (unmatched) parts in $S_B(S_A)$ to each other. The matching outcomes are then fed to the alert correlator to report the final correlation results.

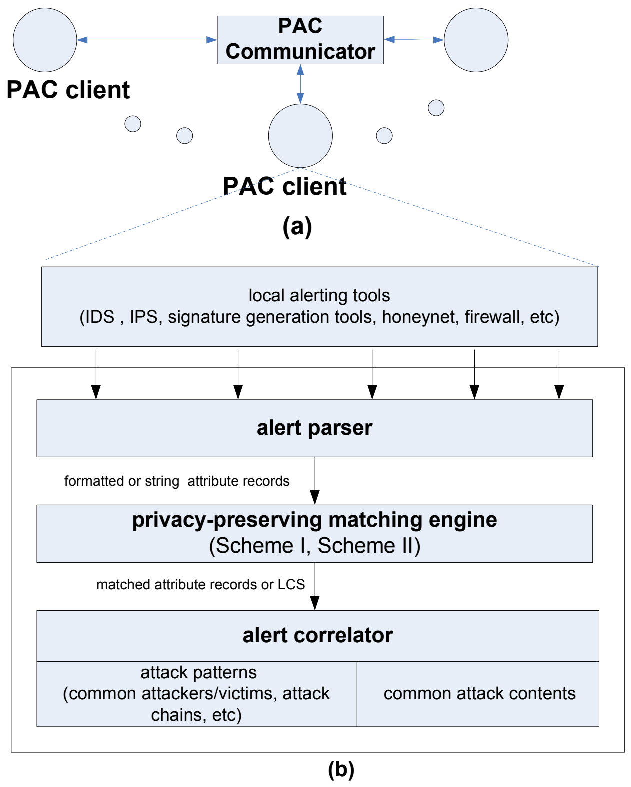

PAC Communicator
PAC client
PAC client
(a)
local alerting tools
(IDS , IPS, signature generation tools, honeynet, firewall, etc)
alert parser
formatted or string attribute records
privacy-preserving matching engine
(Scheme I, Scheme II)
matched attribute records or LCS
alert correlator
attack patterns
(common attackers/victims, attack chains, etc)
common attack contents
(b)

Fig. 11. (a) The PAC system architecture (b) Major modules of a PAC client.

The general workflow of PPME for Protocol II is summarized as follows. First, the PPME of Alice uses an *encryption function E* to map hashed attribute records into ciphertexts, and sends them to the PPME of Bob. The PPME of Bob uses a *modification function M* to create a new ciphertext by bundling the received ciphertexts with his own attribute records, and then returns it back to the PPME of Alice. This way, the PPME of Alice can use a *decryption function D* to compute the matched attribute records of Alice and Bob from the modified ciphertexts. For Protocol VI additional steps are required to calculate the LCS. The terms E (encryption) and D (decryption) are drawn from the naming convention of private set intersection literatures, e.g., [24]. They can be conceptualized as a form of public-private key encryption-decryption chain, except that we obtain the Yes-No results (i.e., two records are identical or not) at end of the process.

In this section we present details of Protocol VI in PPME. We note that the design of Protocol II is based upon the second private set intersection protocol proposed in Section III. C. Protocol VI is designed for Alice and Bob to compute the LCS of strings S_A and S_B, separately owned by them. It is consisted of three phases. In the first phase, it is used to discover common short sub-strings. In the second phase, some (protected) position information of matched short sub-strings is revealed to each other. Finally, longer common sub-strings can be discovered, until the longest common sub-string is found.

In the first phase Alice and Bob agree upon a parameter l, so that both S_A and S_B are broken into two sets of l-grams $L(S_A)$ and $L(S_B)$, respectively. This implies that n-grams whose lengths shorter than l are not considered. In the second phase, positions of matched *l*-grams are privately computed for discovery of all possible common sub-strings, without explicit unveiling of these positions. These positions can be compromised under two conditions: (1) multiple unlinked shorter sub-strings of a user could be used to infer a longer sub-string of the other user, or (2) vice versa. We eliminate these problems by adding randomly generated "faked sub-strings" (or, faked ECC points in our design), such that the protected sub-strings cannot be exhaustively matched by other unlinked shorter sub-strings or a longer sub-string. A randomization function *R* is responsible for generation of those faked points. In the mean time, the randomization should still allow Alice and Bob to discover their adjacent matched *l*-grams within a few rounds of interactions. Selection on the number of faked points is based on the balance between protection of location information and computing cost. In the third phase, Alice (Bob) generates all possible sub-strings based on her (his) common, sequenced *l*-grams. These sub-strings are used for matching by Protocol II to discover the longest common sub-string. Next, we introduce more details about the three phases, and then formally summarize the scheme.

In the first phase, Alice and Bob determine a set of matched *l*-grams $L(C) = \{c_1, c_2, \dots c_w\}$ between $L(S_A)$ and $L(S_B)$ using Protocol II. In the second phase Alice (Bob) first enumerates all sub-strings, $U_A = \{\alpha_1, \alpha_2, \dots, \alpha_{n'}\}$ ($U_B = \{\beta_1, \beta_2, \beta_3, \dots, \beta_{m'}\}$) that can be generated by concatenation of some adjacent c_k in $L_A(L_B)$. Any record in $U_A(U_B)$ is removed if it is a sub-string of some other longer string in $U_A(U_B)$. Next, $\forall\ \alpha_i \in \{\alpha_1, \alpha_2, \dots, \alpha_{n'}\}$, Alice generates *l*-grams $L(\alpha_i) = \{a_{i1}, a_{i2}, \dots\}$. The set of all the *l*-grams generated based on U_A is denoted as $L(U_A) = \{L(\alpha_1), L(\alpha_2), \dots, L(\alpha_{n'})\}$. For each $L(\alpha_i) =$

$\{a_{i1}, a_{i2}, \ldots\} \in L(U_A)$, Alice invokes E of Protocol II to get $E(L(\alpha_i)) = \{E(a_{i1}), E(a_{i2}), \ldots\}$. Then $\forall\, E(L(\alpha_i)) \in \{E(L(\alpha_1)), E(L(\alpha_2)), \ldots, E(L(\alpha_{n'}))\}$, Alice generates randomized-shuffled ciphertext $R(E(L(\alpha_i))$, by inserting random ECC points to $E(L(\alpha_i))$ and then shuffling their order.

The set of randomized-shuffled ciphertexts is denoted as $R(E(L(U_A)) = \{R(E(L(\alpha_1))), R(E(L(\alpha_2))), \ldots , R(E(L(\alpha_{n'})))\}$. Then Alice transforms it into $R(E(L(U_A))'$ by inserting an *infinite point* O of $\mathfrak{E}$ between every two entries in $R(E(L(U_A))$ (i.e., use O as a symbol to separate every two entries in $R(E(L(U_A)))$. Following a similar procedure, Bob also generates $L(U_B) = \{ L(\beta_1), L(\beta_2), \ldots, L(\beta_{m'})\}$ and $R(E(L(U_B))'$. Alice and Bob then exchange $R(E(L(U_A))'$ and $R(E(L(U_B))'$.

With $L(U_A)$ and $R(E(L(U_B))'$ as its inputs, Alice generates $M\{L(U_A), R(E(L(U_B))'\}$ by the modification function M of Protocol II. In the process, the randomization and shuffling and insertion functions mentioned above are also employed. Bob also generates $M\{L(U_B), R(E(L(U_A))'\}$ following a similar procedure. Alice and Bob then exchange their modified ciphertexts $M\{L(U_A), R(E(L(U_B))'\}$ and $M\{L(U_B), R(E(L(U_A))'\}$. Then Alice (Bob) can use D to find $L(U_C)$, which denotes the common l-grams between $L(U_A)$ and $L(U_B)$ with adjacent positions in $U_A(U_B)$. In the last phase, Alice (Bob) generates all possible sub-strings based on $L(U_C)$ *and* $U_A(U_B)$. These sub-strings are used for matching by Protocol II to discover the longest common sub-string.

During execution of E and M, Alice/Bob can protect location information of ciphertexts or modified-ciphertexts by insertion of random ECC points generated by R. For example, in Protocol II when Alice encrypts A to get $E(A) = [E(a_0), E(a_1),\ldots, E(a_n)]$, she could call R to generate additional $2s$ points and attach them with the real ciphertexts, i.e., $R(E(A)) = [E(a_0), E(a_1),\ldots, E(a_n), E(a'_0), E(a'_1),\ldots, E(a'_s)]$, where s is a chosen large integer, $\forall i \in [0, s]$, $E(a'_i) = [E(a'_i)_f, E(a'_i)_b] = (r_A Q, P_i)$, P_i is a randomly generated faked point. R guarantees that Bob cannot tell whether or not a point represents a real attribute record, or a faked point. The detail of Protocol VI is introduced as follows.

Protocol VI. Privacy-Preserving LCS Computation

Input: Alice's string S_A, Bob's string S_B, shortest sub-string length l, and all other security parameters presented in Protocol II

Output: the longest common sub-string LCS between S_A and S_B

(1) *Alice: generate $L(S_A)$ from S_A*

(2) *Bob: generate $L(S_B)$ from S_B*

(3) *Alice $\leftrightarrow$ Bob: Alice and Bob implement Protocol II to find $L(C) = \{c_1, c_2, \ldots c_w\}$, the matched records of $L(S_A)$ and $L(S_B)$*

(4) *Phase 2*

(5) *Alice: generate plaintexts $L(U_A)$ and ciphertexts $R(E(L(U_A))'$ by taking the following steps: generate sub-strings $U_A = \{\alpha_1, \alpha_2, \ldots, \alpha_{n'}\}$, $\forall \alpha_i \in U_A$, α_i is generated from the concatenation of some $c_{a0}, c_{a1}, \ldots, \in L(C)$, where $c_{a0}, c_{a1}, \ldots$ are adjacent in S_A. $\forall \alpha_i \in U_A$, α_i is removed from U_A if there is another $\alpha_i' \in U_A$,α_i is a substring of α_i' for $\alpha_i = \alpha_1$ to $\alpha_i = \alpha_{n'}$, generate l-grams $L(\alpha_i) = \{a_{i1}, a_{i2}, \ldots\}$, the set of all the l-grams of U_A is denoted by $L(U_A) = \{L(\alpha_i), L(\alpha_2), \ldots, L(\alpha_{n'})\}$. For $L(\alpha_i) = L(\alpha_1)$ to $L(\alpha_i) = L(\alpha_{n'})$' suppose $L(\alpha_i) = a_{i1}, a_{i2},\ldots,$, encrypt $a_{i1}, a_{i2},\ldots$ to generate $E(a_{i1}), E(a_{i2}),\ldots$ to get $E(L(\alpha_i)) = \{E(a_{i1}), E(a_{i2})\ldots\}$. For*

$E(L(\alpha_i)) = E(L(\alpha_1))$ to $E(L(\alpha_i)) = E(L(\alpha_{n'}))$, call R to add random points into $E(L(\alpha_i))$ and then shuffle the order of entries in $E(L(\alpha_i))$, the randomized-shuffled ciphertexts is denoted by $R(E(L(\alpha_i))$. Generate $R(E(L(U_A)) = \{R(E(L(\alpha_1)), R(E(L(\alpha_2)), \ldots , R(E(L(\alpha_{n'}))\}$ and transform it into $R(E(L(U_A))'$ by inserting an infinite ECC point O between every two entries in $R(E(L(U_A))$

(6) *Bob: generate $L(U_B)$ and $R(E(L(U_B))'$ following the same steps as Alice*

(7) *Alice $\leftrightarrow$ Bob: exchange $R(E(L(U_A))'$ and $R(E(L(U_B))'$*

(8) *Alice: compute $M(L(U_A), R(E(L(U_B))')$, invoke R and shuffling and insertion function during the modification process, send $M(L(U_A), R(E(L(U_B))')$ to Bob*

(9) *Bob: compute $M(L(U_B), R(E(L(U_A))')$ and send it to Alice following the same procedures*

(10) *Alice: compute $D(M(L(U_A), R(E(L(U_B))'))$, discover all matched l-grams $L(U_C) = \{c_0', c_1', \ldots\}$ and record possible position information, i.e., for any $c_i, c_i' \in L(U_C)$, (a) $c_i, c_i' \in L(\alpha_i)$ & $c_i, c_i' \in R(E(L(\beta_j))$, where $L(\alpha_i) \in L(U_A)$ and $R(E(L(\beta_j)) \in R(E(L(U_B))'$; (b) c_i, c_i' have adjacent positions in α_i*

(11) *Bob: discover matched l-grams using a similar confirmation test*

(12) *Phase 3*

(13) *Alice: generate possible sub-strings based on the common, sequenced l-grams in $L(U_C)$, execute Protocol II with Bob to find the matched sub-string with the longest length*

(14) *Bob: find the matched sub-string with the longest length following the same procedures*

The crypto computing complexity of Protocol VI is that of Protocol II, plus the additional costs required for randomization and shuffling. Both the parameter l and the length of the LCS are important parameters that affect the running time of Protocol VI. For example, if two input strings have no LCS, then only Phase 1 will need to be run, i.e., only one round of matching between two sets of l-grams is required. On the other hand, when two input string have some identical substrings, it is necessary to devise a proportional number of steps for shuffling and randomization (faked points).

In this section we present the proof of the security properties for both schemes. In our design we consider *passive attacks* that aim at compromising confidentiality of sensitive information by analyzing transmitted messages, e.g., dictionary attack. We also consider both *outside adversary* and *inside adversary*: an outside adversary does not participate in the matching process and knows nothing, an inside adversary participates in a matching process who may have some matched records with the targeted victim and try to discover other unmatched records owned by the victim.

Based on the adversary model and attack model introduced above, Protocol II holds *Private Set Intersection* and *Record Confidentiality*, Protocol VI holds *String Confidentiality* and *Partial Sub-String Confidentiality*. (1) Record Confidentiality: either an outside or inside polynomial-time adversary cannot learn what attribute records a peer owns or what matching results two peers obtained by launching any passive attack, if this adversary does not own the same records. Suppose Alice has a set of records A. If $a_i \in A$ and Bob does not have a_i, Bob cannot learn a_i by launching any passive attack. This property holds since in Protocol II large random numbers are used to multiply records to generate *encrypted records* and they are then associated with a fixed ECC point, e.g., $(r_{A1}a_i + r_{A2})Q_A$. We claim that record a_i cannot be deduced from $(r_{A1}a_i + r_{A2})Q_A$ by any passive attack because r_{A1} and r_{A2} are large enough to guarantee the ECDLP. (2) Private Set Intersection: matching peers only discover matched records in the intersection set of their inputs. This property is guaranteed by our matching scheme. Suppose Alice has set A and Bob has set B. They did matching process with each other and found their intersection SI_{AB}. If record $a_i \in A$ and $a_i \notin SI_{AB}$, Bob cannot learn a_i by launching any passive attacks and vice versa. The property holds also because all records are encrypted in a format that guarantees the ECDLP and the detailed analysis is omitted here.

Since Protocol VI is built upon Protocol II, outsiders cannot compromise a user's privacy by eavesdropping transmitted messages, i.e., String Confidentiality is guaranteed. But for malicious insiders, Protocol VI only holds partial confidentiality, named as Partial Sub-String Confidentiality. (3) String Confidentiality: an outside adversary cannot learn what input string a peer owns or what results (the longest common sub-string) two peers obtain. Based on the hardness of ECDLP, any polynomial-time outside adversary, cannot compromise string confidentiality without compromising the string owners. In Protocol VI input strings are processed as *l*-grams, e.g., $L(\quad_i)= \{a_{i1}, a_{i2}, \ldots\}$. All these *l*-grams are matched and transmitted in encrypted format by using Protocol II as the building block. So if Record Confidentiality holds in Protocol II, String Confidentiality also holds for the same reason. (4) Partial Sub-String Confidentiality: an inside adversary cannot learn what sub-strings a peer owns if the insider does not have the same sub-strings. Depending on the randomization function R, the adversary may know what sub-string a peer possibly has if he owns the l-grams that can comprise the substring. That is, if the matched l-grams comprise same sub-strings on both the user's and adversary's side, the adversary will learn this fact; otherwise, the adversary will only know that the user might have the sub-string due to the randomly inserted faked points from R. For example, if Alice calls R to add faked points into $E(A)$, she gets $R(E(A_i)) = [E(a_1), E(a_2),\ldots, E(a_n), E(a'_1), E(a'_2),\ldots, E(a'_s)]$. Here we assume that each record in A represents a l-gram. Since each $E(a'_i) = [E(a'_i)_f, E(a'_i)_b]= (r_AQ, P_i)$ represents a faked encrypted l-gram, where P_i is a randomly generated faked point, even if the first n real l-grams are matched, the adversary only has a small probability to discover the corresponding sub-string by combining them in right sequence from the whole $(s + n)$ l-grams.

A prototype of PAC has been implemented and evaluated it using real alerts from IDS tools such as Snort, worm signature generation tools such as Autograph [92], and network traces collected from a real-world honeynet as its inputs. We used PAC to correlate the alerts for novel applications such as privacy preserving discovery of global attacks, common victims, attack chains, and generation of common malware (e.g., worms/bots) signatures. The software prototype is available at http://creat.tamu.edu/alertcorrelation/. More detail of the evaluation of PAC is presented in Appendix C.

The related work is in three different areas: privacy-preserving alert correlation, alert correlation/aggregation and related cryptographic schemes.

Privacy-Preserving Alert Correlation: One of the first privacy-preserving alert sharing techniques was proposed in [6], which was based on a set of sanitization techniques to hash external IP addresses for matching. It implemented keyed hash functions to encrypt internal IP address and removed other sensitive parts of alerts, e.g., captured and infected data, for privacy protection. The simple and efficient technique supported a few correlation functions. It was vulnerable to dictionary attack. Hash-based sanitization technique was also implemented in [12] to achieve privacy-preserving collaborative intrusion detection based on a P2P networking model. N-gram signature and Bloom filter n-gram signature [13] were proposed for payload-based correlation, which also only supported limited correlation utilities. Another sanitization technique [14] was proposed to generalize sensitive parts of alerts to high-level abstraction, e.g., uncertainty, to be incorporated into the data while retaining partial semantics. The approach was applied to IP addresses and time stamps to discover possible causal relations between attacks. It achieved correlation results on other parts of alerts besides IP addresses, with moderate security support. A semi-centralized architecture was proposed in [23], in which a proxy blinded user inputs and a server identified the blinded keywords by some evaluation function. A survey of previous attempts at privacy-preserving log sharing techniques was presented in [8], e.g., prefix-preserving IP pseudonymizer [93], pseudonym technique [94], packet anonymizer [95] and secure auditing technique [96]. Currently there are some real Internet log and data collection centers such as the Internet Storm Center (ISC) [89] and DShield.org [90]. Most of them are focused on providing centralized applications with good performance, and privacy protection is not their primary concern.

Alert Correlation /Aggregation: Many techniques have been proposed for plaintext alert correlation to discover more attack information. A formal data model was proposed IDS alert correlation in [97]. A decision-theoretic alert fusion technique [98] was proposed to improve the performance of combination of diverse intrusion detectors. Another approach [99] was presented to construct attack scenarios by correlating alerts based on the prerequisites and consequences of intrusions. In [100] a tool was proposed to perform real-time alert verification. Another alert aggregation and correlation algorithm [101] was presented to discover a more condensed view of alerts. All these alert correlation techniques require matching of certain (plaintext) attributes, and our technique can be viewed as a supporting tool when privacy protection is a major concern for distributed alert collaborators.

Related cryptographic schemes: Most of the related have been discussed in Section III. The main reason that there have not been any practical privacy-preserving alert correlation tools is because that the high computational complexity of most previous PSI or SMC protocols are major hurdles for their practical use in correlation of large volume of inputs [23][91].

VII. SUMMARY

The objective of this research work is to develop a general framework for design of efficient and secure privacy-preserving element matching and credential authentication protocols, and implement the protocols on various Internet applications with privacy protection requirement. While most existing cryptographic protocols/algorithms face challenges of poor efficiency, high computing costs, and inadequate security protections for different kinds of critical Internet application with high security and performance requirement, we proposed two private set intersection (PSI) protocols with linear computing and communication costs, an unlinkable secret handshake (SH) protocol with reusable credential and least communication cost, and interlocked protocols with new security properties. The proposed protocols were also implemented to design a privacy-preserving inquiry matching system for online social applications and a privacy-preserving alert correlation system for collaboration of different sources of network alerts.

A general framework was presented to illustrate the mathematical basis under privacy-preserving element matching and credential authentication protocols. A homomorphic randomization function was proposed to meet the conflicting goals: matched (verified) elements (credentials) are computed through homomorphism, while unmatched (unverified) elements (credentials) are protected through randomization. Based on the function we presented common requirement between PSI and SH protocols, and the three common phases of PSI and SH protocols: Protocol Initialization phase, Secret Matching phase and Result Computing phase. In addition, the general attack model and adversary model for both PSI and SH protocols were introduced in this dissertation.

PSI protocols are used for privacy-preserving element matching. The existing PSI solutions have $O(n^2)$ or $O(nL)$ computing cost, where n is the number of elements and L is the average bit length of an element. Based on the general framework we proposed two new PSI protocols with linear computing cost $O(n)$. The first protocol uses full homomorphic randomization function as the cryptographic basis, and achieves element confidentiality and private set intersection in honest-but-curious adversary model with linear computing cost. The second protocol uses partial homomorphic randomization function as the cryptographic basis. Besides element confidentiality and private set intersection, the second protocol is also more secure against malicious attacks like falsification.

SH protocols are used for privacy-preserving credential authentication. The existing SH solutions either requires a large number of pairs of single-use credentials and pseudonyms or *6* bilinear mapping operations to achieve unlinkability, which is one of the most important security properties of SH protocols. Based on the general framework we proposed a new SH protocol that achieves unlinkability with a reusable pair of credential and pseudonym. Our solution only requires *2* bilinear mapping operations. In our design a party directly chooses random numbers and associates them with his pseudonym to minimize the correlation between his credentials and pseudonyms.

Based on the proposed framework we proposed to interlock PSI protocols and SH protocols to design two protocols with new security properties. In the first protocol a PSI protocol is executed first and then the matched elements are associated with the credentials in a following SH protocol. The first interlocked protocol guarantees authenticity on matched elements. In the second protocol a SH protocol is executed first and then the pairwise session key and verified credentials are utilized in a following PSI protocol. The second interlocked protocol guarantees detection resistance and impersonation attack resistance on matching elements.

We implemented the proposed PSI and SH protocols to provide privacy-preserving inquiry matching service (PPIM) for social networking applications and privacy-preserving correlation service (PAC) of network security alerts. In PPIM a system based on an end-user privacy control model is built, in which online social service consumers first use non-privileged information to discover potential candidates and then engage in PPIM transactions with the candidates to exchange encrypted inquiries. The proposed PSI protocol is called in PPIM for matching of (encrypted) inquires and the proposed SH protocol is called in PPIM for verification of group memberships (credentials). In the end only the owner and her peer with matched inquiries and group memberships can know the real transaction contents, but not any other users, including our PPIM server. In PAC a system is built to correlate network security alerts between semi-honest users. Two different protocols based on the homomorphism of the elliptic curve cryptosystem (ECC) are implemented for PAC. The first protocol is the proposed partial homomorphism based PSI protocol, which is implemented for privacy-preserving matching of alerts with known formats, e.g., IP addresses, port numbers, etc. Expanding from the matching protocol, the second protocol is designed for privacy-preserving computation of the longest common sub-string (LCS) between two input strings, which is used for generating common contents of captured malware/intrusion payloads.

REFERENCES

[1] R. Gross, A. Alessandro, and H. Heinz, “Information revelation and privacy in online social networks,” in *Proc. ACM Workshop on Privacy in the Electronic Society*, Nov. 2005, pp. 71-80.

[2] S. Guha, K. Tang, and P. Francis, “NOYB: privacy in online social networks,” in *Proc. WOSN*, Aug. 2008, pp. 49-54.

[3] M. Chew, D. Balfanz, and B. Laurie, “(Under)mining privacy in social networks,” in *W2SP* workshop, May 2008, pp. 45-49.

[4] A. Korolova, R. Motwani, S. Nabar, and Y. Xu, “Link privacy in social networks,” in *Proc. ICDE*, Oct. 2008, pp. 1355-1357.

[5] M. M. Lucas and N. Borisov, “flyByNight: mitigating the privacy risks of social networking,” in *WPES* workshop, Oct. 2008, pp. 1-8.

[6] P. Lincoln, P. Porras and V. Shmatikow, “Privacy-preserving sharing and correlation of security alerts,” in *Proc. USENIX*, 2004, pp. 239-254

[7] P. Porras and V. Shmatikov, “Large-scale collection and sanitization of network security data: risks and challenges,” in *NSPW* workshop, Sep. 2006, pp. 57-64.

[8] A. Slagell and W. Yurcik, “Sharing computer network logs for security and privacy: a motivation for new methodologies of anonymization,” in *Proc. SECOVAL*, Sep. 2005, pp. 80-89.

[9] J. Bethencourt, J. Franklin and M. Vernon, “Mapping Internet sensors with probe response attacks,” in *Proc. USENIX Security Symposium*, 2005, pp. 193-208.

[10] R. Wang, X. Wang, Z. Li, H. Tang, M. Reiter and Z. Dong, “Privacy-preserving genomic computation through program specialization,” in *Proc. CCS,* Nov. 2009, pp. 338-347.

[11] R. Wang, Y. Li, X. Wang, H. Tang and X. Zhou, “Learning your identity and disease from research papers: information leaks in genome wide association study,” in *Proc. CCS,* Nov. 2009, pp. 534-544.

[12] M. Locasto, J. Parekh, A. Keromytis, and S. Stolfo, “Towards collaborative security and P2P intrusion detection,” in *Proc. IEEE Information Assurance Workshop,* Jun. 2005, pp. 333–339.

[13] J. J. Parekh, K. Wang, S. J. Stolfo, "Privacy-preserving payload-based correlation for accurate malicious traffic detection," in *Proc. SIGCOMM Workshop on Large Scale Attack Defense*, 2006, pp. 99-106.

[14] D. Xu and P. Ning, "Privacy-preserving alert correlation: a concept hierarchy based approach," in *Proc. Computer Security Applications Conference*, Dec. 2005, pp. 537-546.

[15] M. Naor, B. Pinkas, and R. Sumner, "Privacy preserving auctions and mechanism design," in *ACM Conf. on Electronic Commerce*, 1999, pp. 129-139.

[16] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *Proc. ACM SIGMOD*, 2000, pp. 439–450.

[17] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," *Journal of the ACM,* vol. 45, no. 6, pp. 965-982, 1998.

[18] S. Jha, L. Kruger, and V. Shmatikov, "Towards practical privacy for genomic computation," in *Proc. IEEE Symposium on Security and Privacy*, May 2008, pp. 216-230.

[19] A. Leung and C.J. Mitchell, "Ninja: non identity based, privacy preserving authentication for ubiquitous environments," in *Proc. International Conference on Ubiquitous Computing*, 2007, pp. 73–90.

[20] L. Lu, J. Han, L. Hu, Y. Liu, and L. M. Ni, "Dynamic key-updating: privacy-preserving authentication for RFID systems," in *Proc. Pervasive Computing and Communications*, Mar. 2007, pp. 13–22.

[21] P. Porras and V. Shmatikov, "Large-scale collection and sanitization of network security data: risks and challenges," in *Proc. New Security Paradigms Workshop*, Sep. 2006, pp. 57-64.

[22] W. Du and M. J. Atallah, "Secure multi-party computation problems and their applications: a review and open problems," in *Proc. New Security Paradigms Workshop*, Sep. 2001, pp. 11-20.

[23] H. Ringberg, B. Applebaum, M. J. Freedman, M. Caesar and J. Rexford, "Collaborative privacy-preserving data aggregation at scale," in Cryptology ePrint Archive, Report 2009/180, 2009.

[24] M. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *Proc. EUROCRYPT*, May 2004, pages 1-19.

[25] D. Balfanz, G. Durfee, N. Shankar, D. Smetters, J. Staddon, and H. Wong, “Secret handshakes from pairing-based key agreements,” in *Proc. IEEE Symposium on Security and Privacy*, May 2003, pp.180-196.

[26] S. Garriss, M. Kaminsky, M. J. Freedman, B. Karp, D. Mazieres, and H. Yu, “RE: reliable email,” in *Proc. NSDI* , May 2006, pp. 297-310.

[27] C. Hazay and Y. Lindell, “Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries,” in *Proc. TCC*, Mar. 2008, pp. 155-175.

[28] G. Ateniese and M. Blanton, “Secret handshakes with dynamic and fuzzy matching,” in *Proc. NDSS*, Feb. 2007, pp. 159-177.

[29] “Facebook.” Available: http://www.facebook.com/

[30] “Snort.” Available: http://www.snort.org/

[31] H. Kim and B. Karp, “Autograph: toward automated, distributed worm signature detection,” in *Proc. USENIX Security Symposium*, Aug. 2004, pp. 271-286.

[32] L. Kissner and D. Song, “Privacy-preserving set operations,” in *Proc. Advances in Cryptology-CRYPTO*, Aug. 2005, pages 241–257.

[33] J. Camenisch and G. M. Zaverucha, “Private intersection of certified sets,” in *Proc. Financial Cryptography*, Feb. 2009, pp. 108–127.

[34] S. Jarecki and X. Liu, “Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection,” in *Proc. TCC*, Mar. 2009, pp. 577–594.

[35] E. De Cristofaro, J. Kim, and G. Tsudik, “Linear-complexity private set intersection protocols secure in malicious model,” in *Proc. ASIACRYPT*, Dec. 2010, pp. 213-231.

[36] J. S. Shin and V. D. Gilgor, “A new privacy-enhanced matchmaking protocol,” in *Proc. NDSS Symposium*, Feb. 2008.

[37] J. Katz, R. Ostrovsky, and M. Yung, “Efficient password-authenticated key exchange using human-memorable passwords,” in *Proc. EUROCRYPT*, May 2001, pp. 475-494.

[38] Y. Sang, H. Shen, Y. Tan, and N. Xiong, "Efficient protocols for privacy preserving matching against distributed datasets," in *Proc. ICICS*, Dec. 2006, pp. 210 – 227.

[39] Y. Sang and H. Shen, "Privacy preserving set intersection based on bilinear groups," in *Proc. ACSC*, Jan. 2008, pp. 47-54.

[40] F. Kerschbaum, A. Schaad and D. Biswas, "Practical privacy-preserving protocols for criminal investigations," in *Proc. IEEE Intelligence and Security Informatics*, Jun. 2009, pp. 197-199.

[41] A. C. Yao, "Protocols for secure computations," in *Proc. IEEE Symposium on Foundations of Computer Science*, Nov. 1982, pp. 160-164.

[42] D. Malkhi, N. Nisan, B. Pinkas and Y. Sella, "Fairplay – a secure two-party computation system," in *Proc. USENIX Security Symposium*, Aug. 2004, pp. 287-302.

[43] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987.

[44] V. Miller, "Use of elliptic curves in cryptography," in *Proc. CRYPTO*, 1985, pp. 417-426.

[45] L. C. Washington, *Elliptic Curves: Number Theory and Cryptography*. Boca Raton, FL: Chapman & Hall/CRC, 2003.

[46] S. Jarecki and X. Liu, "Unlinkable secret handshakes and key-private group key management schemes," in *Proc. Applied Cryptography and Network Security*, Jun. 2007, pp. 270-287.

[47] "Opensocial." Available: http://www.opensocial.org/

[48] "Match.com." Available: http://www.match.com/matchus/

[49] "LinkedIn." Available: http://www.linkedin.com/

[50] "ParterUp." Available: http://www.partnerup.com/

[51] "Sermo.com." Available: http://www.sermo.com/

[52] "INmobile.org." Available: http://www.inmobile.org/information/

[53] R. Konrad, "Facebook opens to third-party developers," Available: http://www.msnbc.msn.com/id/18899269/.

[54] J. E. Vascellaro, "Social networking goes professional," *THE WALL STREET JOURNAL*. Available: http://online.wsj.com/article/SB118825239984310205.html.

[55] "Facebook Privacy Policy." Available: http://www.facebook.com/ policy.php.

[56] "Opensocial." Available: http://www.opensocial.org/.

[57] "OpenID." Available: http://openid.net.

[58] "The DataPortability project." Available: http://dataportability.org.

[59] A. Narayanan and V. Shmatikov, "De-anonymizing social networks," in *Proc. IEEE S&P*, May 2009, pp. 173-187.

[60] WellNet Social Networking. Available: http://www.wellnethealthcare. com/.

[61] B. Popescu, B. Crispo, and A. Tanenbaum, "Safe and private data sharing with Turtle: Friends team-up and beat the system," in *Proc. Cambridge Workshop on Security Protocols*, Apr. 2004, pp. 213-220.

[62] A. Felt and D. Evans, "Privacy protection for social networking platforms," in *W2SP* workshop, May 2008, pp. 37-44.

[63] E. Mills, "Facebook suspends app that permitted peephole," Available: http://news.cnet.com/8301-10784 3-9977762-7.html.

[64] M. Arrington, "Don't post the evidence unless it supports your case," Available: http://tinyurl.com/6otok7.

[65] R. Gross, A. Alessandro, and H. Heinz, "Information revelation and privacy in online social networks," in *Proc. ACM Workshop on Privacy in the Electronic Society*, Nov. 2005, pp. 71-80.

[66] L. Backstrom, C. Dwork, and J. Kleinberg, "Wherefore art thou R3579X? anonymized social networks, hidden patterns, and structural steganography," in *Proc. WWW*, May 2007, pp. 181-190.

[67] "Privacy and security issues in social networking," Fastcompany, 2008. Available: http://www.fastcompany.com/articles/2008/10/social-networking-security.html.

[68] P. Bearman, J. Moody, and K. Stovel, "Chains of affection: the structure of adolescent romantic and sexual networks," *American Journal of Sociology*, vol. 100, no. 1, pp. 44-91, 2004.

[69] "A collection of social network stats for 2009," Jeremiah Owyang, 2009. Available: http://www.web-strategist.com/blog/2009/01/11/a-collection-of-soical-network-stats-for-2009/.

[70] N. O'Neill, "Senate begins discussing privacy implications of online advertising," 2008. Available: http://tinyurl.com/5aqqhe.

[71] E. Eldon, "VentureBeat: MediaSixDegrees targets ads using social graph information," 2008. Available: http://tinyurl.com/662q3o.

[72] K. B. Frikken and P. Golle, "Private social network analysis: how to assemble pieces of a graph privately," in *Proc. Workshop on Privacy in Electronic Society*, Oct. 2006, pp. 89-97.

[73] B. Laurie, "Apres: a system for anonymous presence," 2004. Available: http://www.apache-ssl.org/apres.pdf.

[74] K. Liu and E. Terzi, "Towards identity anonymization on graphs," in *Proc. SIGMOD*, Jun. 2008, pp. 93-106.

[75] B. Zhou and J. Pei, "Preserving privacy in social networks against neighborhood attacks," in *ICDE*, Apr. 2008, pp. 506-515.

[76] E. Zheleva and L. Getoor, "Preserving the privacy of sensitive relationships in graph data," in *PinKDD*, Apr. 2007, pp. 153-171.

[77] R. Baden, A. Bender, N. Spring, B. Bhattacharjee and D. Starin, "Persona: an online social network with user-defined privacy," in *SIGCOMM*, Aug. 2009, pp. 135-146.

[78] "Patientslikeme." Available: http://www.patientslikeme.com/

[79] "Man-in-the-middle attack." Available: http://en.wikipedia.org/wiki/ Man-in-the-middle attack.

[80] Y. Zhang, W. Liu and W. Lou, "MASK: anonymous on-demand routing in mobile ad hoc networks," *IEEE Trans. on Wireless Communications*, vol. 5(9), pp.2376-2385, 2006.

[81] J. Douceur, "The sybil attack," in *Proc. IPTPS*, Mar. 2002, pp. 251-260.

[82] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "SybilGuard: defending against sybil attacks via social networks," *IEEE Trans. on Networking*, vol. 16, no. 3, pp. 576-589, 2008.

[83] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao, "SybilLimit: a near-optimal social network defense against sybil attacks," *IEEE Trans. on Networking*, vol. 18, no. 3, pp. 885-898, 2010.

[84] G. Danezis and P. Mittal, "SybilInfer: detecting sybil nodes using social networks," in *Proc. NDSS*, Feb. 2009.

[85] S. Galbraith, K. Harrison and D. Soldera, "Implementing the tate pairing," in *Proc. Algorithm Number Theory Symposium*, Jul. 2002, pp. 324-337.

[86] "MIRACL Library." Shamus Software Ltd. Available: http://indigo.ie/~mscott/.

[87] "US Secure Hash Algorithm 1 (SHA-1)." Available: http://tools.ietf.org/html/rfc3174.

[88] J. Bethencourt, J. Franklin and M. Vernon, "Mapping Internet sensors with probe response attacks," in *Proc. USENIX*, Aug. 2005, pp. 193-208.

[89] "ISC – Internet Storm Center." Available: http://isc.incidents.org, 2005.

[90] "Distributed Intrusion Detection System," DShield.org. Available: http://www.dshield.org, 2005

[91] U. Meyer, S. Wetzel and S. Ioannidis, "New advances on privacy-preserving policy reconciliation," in Cryptology ePrint Archive, Report 2010/064, 2010.

[92] H. Kim and B. Karp, "Autograph: toward automated, distributed worm signature detection," in *Proc. USENIX*, Jun. 2004, pp. 271-286.

[93] J. Xu, J. Fan, M. H. Ammar and S. B. Moon, "On the design and performance of prefix-preserving IP traffic trace anonymization," in *Proc. ACM SIGCOMM Internet Measurement Workshop*, Nov. 2001, pp. 263-266.

[94] J. Biskup and U. Flegel, "Transaction-based pseudonyms in audit data for privacy respecting intrusion detection," in *Proc. Recent Advances in Intrusion Detection*, Oct. 2000, pp. 28-48.

[95] R. Pang and V. Paxson, "A high-level programming environment for packet trace anonymization and transformation," in *Proc. ACM SIGSOMM*, Aug. 2003, pp. 339-351.

[96] B. Waters, D. Balfanz, G. Durfee and D.K. Smetters, "Building an encrypted and searchable audit log," in *Proc. Internet Society Network Distributed Systems Symposium*, Jan. 2004, pp. 205-214.

[97] B. Morin, L. Me, H. Debar, and M. Ducasse, "M2D2: a formal data model for IDS alert correlation," in *Proc. Recent Advances in Intrusion Detection*, Oct. 2002, pp. 115-137.

[98] G. Gu, A. A. Cardenas, and W. Lee, "Principled reasoning and practical applications of alert fusion in intrusion detection systems," in *ASIACCS*, Mar. 2008, pp. 136-147.

[99] P. Ning, Y. Cui, and D. Reeves, "Constructing attack scenarios through correlation of intrusion alerts," in *Proc. ACM CCS*, Nov. 2002, pp. 245-254.

[100] C. Kruegel, G. Vigna, W. Robertson, "Using alert verification to identify successful intrusion attempts," *Journal of Practice in Information Processing and Communication*, vol. 27, no. 4, pp. 220-228, 2004.

[101] H. Debar and A. Wespi, "Aggregation and correlation of intrusion-detection alerts," in *Proc. International Symposium on Recent Advances in Intrusion Detection*, Oct. 2001, pp. 85-103.

[102] W. Mao. *Modern cryptography: theory and practice*. NJ: Prentice Hall, 2003.

[103] "Wireshark." Available: http://www.wireshark.org/.

[104] M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," in *Proc. CCS*, Nov. 1993, pp.62-73.

[105] M. Bellare and P. Rogaway, "The exact security of digital signatures how to sign with RSA and Rabin," in *Proc. EUROCRYPT*, Mar. 1996, pp.399-416.

[106] D. Pointcheval and J. Stern, "Security proofs for signature schemes," in *Proc. EUROCRYPT*, Mar. 1996, pp.387-398.

[107] D. Pointcheval and J. Stern, "Provable secure blind signature schemes," in *Proc. ASIACRYPT*, Nov. 1996, pp.252-265.

[108] NIST, "Secure Hash Standard (SHS)," FIPS PUB 180-2, Aug. 2002.

[109] R. Canetti, O. Goldreich and S. Halevi, "The random oracle methodology, revisited," *Journal of the ACM*, vol. 51, no. 4, pp. 557-594.

[110] D. Boneh and X. Boyen, “Efficient selective-ID secure identity based encryption without random oracles,” in *Proc. EUROCRYPT*, May 2004, pp.223-238.

[111] D. Boneh and X. Boyen, “Short signatures without random oracles,” in *Proc. EUROCRYPT*, May 2004, pp.56-73.

[112] A. Menezes, T. Okamoto and S. Vanstone, “Reducing elliptic curve logarithms to logarithms in a finite field,” *IEEE Transaction on Information Theory*, vol. 39, no. 5, pp.1639-1646, 1993.

[113] G. Frey and H. G. Ruck, “A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves,” *Mathematics of Computation*, vol. 62, no. 206, pp.865-874, 1994.

[114] A. Miyaji, M. Nakabayashi, and S. Takano, “New explicit conditions of elliptic curve traces for FR-reduction,” *IEICE Transactions on Fundamentals*, vol. E84-A, no. 5, pp. 1234-1243, 2001.

[115] H. Cohen, G. Frey and R. Avanzi. *Handbook of Elliptic Curve and Hyperelliptic Curve Cryptography*. Boca Raton, FL: Chapman & Hall/CRC, 2006.

APPENDIX A

CRYPTOGRAPHIC PRIMITIVES

Group

Let $\{0, 1\}$ denote the set of individual bits and let $\{0, 1\}^*$ denote the set of all bit strings. $N = \{0, 1, 2, \ldots\}$ is the set of all natural numbers. $\kappa \in N$, 1^κ and $\{0, 1\}^\kappa$ denotes the bit string of κ ones and the set of bit strings of length κ, respectively. When a and b are strings, then $|a|$ refers to the length of the string a and $a||b$ indicates the concatenation of a and b. In this proposal "group" and "field" refer to the *group* and *field* defined in number theory. Abelian groups and finite fields are two important algebraic structures in modern cryptography. They are widely used in real world applications because they provide the basic elements and operations for cryptographic schemes. A group is thought as a set of objects with an operation between two elements in this set. It can be described by mathematical language as follows [102]:

Definition A. I: a group $(G, \lozenge)$ is a set G together with an operation $\lozenge$ which satisfies the following conditions: For all a, $b \in G$, $a \lozenge b \in G$. For all a, b, $c \in G$, $a \lozenge (b \lozenge c)=(a \lozenge b) \lozenge c$.There exists a unique element $d \in G$, for all $a \in G$, $a \lozenge d = d \lozenge a = a$. This element d is called the identity element. For all $a \in G$, there exists a^{-1} such that $a \lozenge a^{-1} = d$

Public Key Cryptography

Digital signatures are one of the most important cryptographic tools in modern cryptography. In Public Key Cryptography (PKC), every user has two keys: One is the public key which is published to everyone, and the other is the secret (or private) key which is kept secret by the corresponding owner. In a traditional PKC, the key pair is generated by users or by a key generator called Certificate Authority (CA). In identity-based PKC (ID-PKC), the key pair is generated by a key generator called Trusted Authority (TA). It is worthy of noting that their roles are different. When the key pairs are generated by users, the CA does not know the users' secret key and just issues the certificate with the binding of identities and public keys. The TA checks that applicants have claimed identity and then issues the corresponding secret key. Thus ID-PKC has a key escrow facility. Whether this facility is useful or not depends on the particular applications. Both PKC and ID-PKC use the difficulty of the mathematical problems, such as integer factoring problems and discrete logarithm problems, to guarantee that no one is able to recover the secret (private) key from the public key. The difference between PKC and ID-PKC is that the public key is random in PKC, while the public key in ID-PKC is specified before generating the private/secret key.

Random Oracle Model and Standard Model

The hash function is a very useful tool in modern cryptography because it has the property of collision-resistant. Collision-resistant means that it is impossible for anyone to find two different messages providing the same hash value. The hash function was originally designed to be used in digital signature schemes for signing long messages by reducing the size of the signatures without comprising the integrity of the message. In order to obtain related arguments in the security proof of cryptographic schemes, some cryptographers [104][105][106][107] have suggested that hash functions should be seen as random functions. Therefore, a new model, the random oracle model (ROM), has been introduced into the security proofs of cryptographic schemes. If one expects to prove the security of a cryptographic scheme where hash functions are used, then these hash functions are viewed as random oracles. This often makes it easy and simple to prove the security of cryptographic schemes. The random oracles will return a truly random value for each new query from a challenger. If a new query is the same as a previous query, the random oracle will return an identical answer. This theoretical model can help cryptographers verify whether the design of a cryptographic scheme is secure or not. However, the random oracle model is only an imaginary model in theory. There does not exist a true random oracle in the real world. In real world applications, hash functions such as SHA-224 [108] play the role of random oracles. Since the concept of the random oracle model was introduced into the community, researchers have used it to provide the security proof for numerous cryptographic schemes. Although many cryptographic schemes have been proven in the random oracle model, some researchers doubted the reliability of this model. As what has been discussed in previous section, hash functions are usually treated as random imaginary functions when proving the security of the cryptographic schemes in the random oracle model. Recently, Canetti, Goldreich and Halevi [109] investigated the reliability of the random oracle and presented some useful results. They showed that there were some examples of cryptographic schemes which are secured under the random oracle model but not in real applications. Therefore, constructing cryptographic schemes without using random oracles, namely in the standard model, has become an active research area. Encryption schemes and signature schemes that do not use random oracles have been proposed

[110][111]. Their security has been proven in the standard model. It is noted that, although the authors of [109] have shown the weakness of the random oracle model, they did not agree that the random oracle model was of no use in proving the security of the cryptographic schemes. Until now, there has been no explicit conclusion that cryptographic schemes should not use random oracles. Therefore, new cryptographic schemes with random oracles whose security is also proven in the random oracle model are still being constructed.

Elliptic Curve Cryptosystem

Elliptic curve cryptography (ECC) was proposed by Koblitz [43] and Miller [44]. They independently suggested the use of elliptic curve groups in Public Key Cryptography. Finite abelian groups are fundamental to Public Key Cryptography. Elliptic curves over finite fields offer a large number of such finite abelian groups. In the past twenty years, researchers have paid much attention to ECC and developed many important cryptographic schemes. Compared with cryptosystems based on RSA or DLP, ECC has several advantages. One of them is that ECC is able to provide reliable security with shorter key lengths than other cryptosystems (more). Next, we review the definition of elliptic curves over finite fields. Let $K = F_p$ be a finite field. Suppose $E(K)$ is an elliptic curve E over a field K, it is defined as the set of points (x, y) with x, $y \in K$ that satisfy the Weierstrass equation. $E(K)$ also includes an extra point O called point at infinity which is usually denoted by $O =(x, 1)$. The number of points on $E(K)$ is called the order of $E(K)$ which is denoted by $E(K)$ or $\# E$. When $K = F_p$ for a large prime $p \geq 5$, the Weierstrass equation shown in Equation 2.1 is simplified into the following equation

$y^2 = x^3 + a_4x + a_6$. An elliptic curve is the set of points (x,y) which satisfy the equation $y^2 = x^3 + a \cdot x + b$, where x, y, a, and b are over same finite field, e.g. F_p. If $4 \cdot a3 + 27 \cdot b2 \neq 0$ (i.e. $x3 + a \cdot x + b$ contains no repeated factors), then points on an elliptic curve can be used to form an additive group, usually written as $E(F_p)$. Addition and doubling are the two operations defined on a group of curve points. They are further used to compute *point multiplication*, which is the basis of any elliptic curve cryptographic schemes: given an integer k, computation of kP is called point multiplication. It was based on a famous hard problem, i.e., Elliptic Curve Discrete Logarithm Problem. Given points $P, Q \in E(F_q)$ ($E(F_q)$ is the additive group formed by the points on an elliptic curve), finding an integer $k \in F_q$ such that $k \cdot P = Q$ is called Elliptic Curve Discrete Logarithm Problem.

Definition A. II: for an elliptic curve E over a finite field K, where P, Q are two points on $E(K)$, it is difficult to determine $\lambda \in Z$ such that $Q = \lambda P$

Pairing-Based Cryptography

Menezes, Okamoto and Vanstone [112] proposed an efficient attack method for supersingular elliptic curves called MOV attack. That is, an ECDLP on supersingular elliptic curves can be reduced to a much easier DLP in a finite field by using bilinear pairings, such as Weil pairing [112] or Tate pairing [113]. Therefore, such a supersingular elliptic curve was seen as weak and to be avoided when selecting elliptic curve parameters. Later the idea was implemented to generate pairing-based cryptosystem [114]. Let q be a positive integer. G_1 and G_2 are two additive groups of order q where DLP is difficult. G_3 is a multiplicative group of order q. A bilinear pairing e can be defined as $e : G_1 \times G_2 \rightarrow G_3$

Currently, the bilinear pairings used in most existing cryptographic schemes are derived from Tate pairings or Weil pairings, which can be implemented on a supersingular elliptic curve or a suitable non-supersingular elliptic curve. G_1 and G_2 are usually regarded as two Abelian groups constructed by elliptic curves over finite fields. Let k be an integer larger than 1. In the following description, k is either a finite field F_p^k or a subfield of F_p^k, and k is defined as an embedding degree or security parameter. The bilinear pairings in ECC are usually thought of as functions which map two elliptic curve points to an element of a multiplicative group. Let O denote the identity element of G_1 or G_2 and let 1 denote the identity element of G_3. The bilinear pairings have the following properties:

Definition A. III Bilinearity: For any $P_1, P_2 \in G_1$, $Q_1, Q_2 \in G_2$, $e(P_1 + P_2; Q_1)= e(P_1; Q_1)\, e(P_2; Q_1)$ and $e(P_1; Q_1 + Q_2)= e(P_1; Q_1)\, e(P_1; Q_2)$.

Definition A. IV Non-degeneracy: For all $P \in G_1$ except O, there is some point $Q \in G_2$ such that $e(P, Q) = 1$. In the same way, for all $Q \in G_2$ except O, there is some point $P \in G_1$ such that $e(P, Q) \neq 1$.

Suppose G_1 denotes an additive cyclic group of prime order q. G_2 denotes a multiplicative cyclic group of order q. G_1 and G_2 are selected in such a way that the *Discrete Logarithm Problem* (DLP) [45] is hard in both of them. Let G be an additive group. Computational Diffie-Hellman Problem (CDHP) in G is defined by

Definition A. V: Given P , aP and $bP \in G$ for a, $b \in Z^*_p$, compute abP.

Decisional Diffie-Hellman Problem (DDHP) [115] in G is defined by

Definition A. VI*:* Given P , aP , bP and $cP \in G$ for a, b and $c \in Z^*_p$, decide whether $c \equiv ab \ (mod\ p)$ or not. If true, (P , aP , bP , cP) is called a Diffie Hellman tuple.

Bilinear Diffie-Hellman (BDH) [115] Assumption is defined by

Definition A. VII*:* given P, aP, bP, cP for random a, b, $c \in Z^*_q$ and $P \in G_1$, it is not possible to compute $e(P, P)^{a \cdot b \cdot c}$ with a non-negligible probability, i.e., it is hard to compute $e(P, P)^{a \cdot b \cdot c}$.

A group G with prime order is called Gap-Diffie-Hellman (GDH) group if DDHP can be solved in polynomial time while no known polynomial time algorithm can solve CDHP with non-negligible advantage in polynomial time. An additive group G constructed on an elliptic curve over finite fields is such a GDH group due to the existence of the bilinear pairings. MOV attack [112] based on Weil pairing and FR attack [113] based on Tate pairing make it possible to solve DDH in G, whereas there is no known polynomial time algorithms for CDHP in G. CDHP and DDHP are the fundamentals of pairing-based cryptosystems. These mathematical assumptions will be used widely in advanced pairing-based cryptographic schemes.

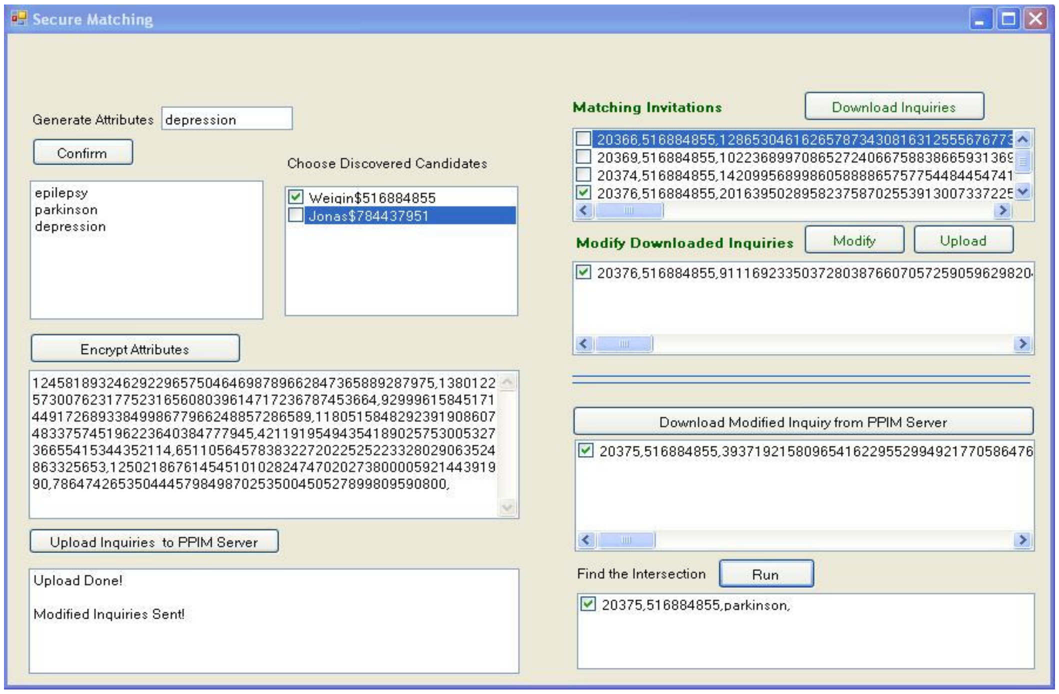
APPENDIX B

EVALUATION OF PPIM

To demonstrate the utility of the PPIM service, we developed a patient support community called “PatientMatch”, using Facebook as the SNS. The objective of the PatientMatch prototype is to form a supporting group among qualified constituents (patients, social workers, etc), for a particular type of health conditions (chronic illness, substance abuse, etc) to simulate the type of services provided in [78], but keep other people out of the group. Fig. 12 (a) shows a snapshot of the user interface of the PatientMatch client tool, which only included the PSI functions. We have also implemented the SH protocol, but it was only used for the running time evaluation.

Facebook platform assigns API Key and Application Secret for running each application. Facebook uses callback URL to forward the request to users between applications web pages. We implemented PatientMatch as a desktop application in order to fully support the crypto-related computations. Otherwise, the web-based application would be subject to the restriction of the javascript.

The PPIM server is implemented as a web service using standard HTTP port to exchange XML messages with PPIM client, based on the Simple Object Access Protocol (SOAP). The software architecture for the PPIM server and its client tool is shown in Fig. 12 (b). A prototype of the PPIM server and our Patient- Match sample desktop application can be accessed through “http://apps.facebook.com/secretsharesearch”. The PPIM client is compiled into a DLL with an exposed API so that it can be used as a plug-in by existing SNS applications. Following an asynchronous messaging model, the PPIM server uses a (MySql) database to buffer any unprocessed (ciphertext) messages between principals. The PPIM client provides all the cryptographic functions (encryption, injection and modification of ciphertexts, and decryption) and PPIM related networking functions.


Secure Matching
Confirm
Encrypt Attributes
Upload Inquiries to PPIM Server
Download Inquiries
Modify
Upload
Download Modified Inquiry from PPIM Server
Run

Generate Attributes depression

epilepsy
parkinson
depression

Choose Discovered Candidates

✓ Weiqin$516884855
☐ Jonas$784437951

```
1245818932462922965750464698789662847365889287975,1380122
573007623177523165608039614717236787453664,92999615845171
4491726893384998677966248857286589,1180515848292391908607
4833757451962236403847779945,42119195494354189025753005327
36655415344352114,651105645783832272022525223328029063524
863325653,12502186761454510102824747020273800005921443919
90,786474265350444579849870253500450527899809590800,
```

Upload Done!

Modified Inquiries Sent!

Matching Invitations

☐ 20366,516884855,12865304616265787343081631255567677
☐ 20369,516884855,10223689970865272406675883866593136
☐ 20374,516884855,142099568998605888865757754484454741
✓ 20376,516884855,20163950289582375870255391300733722

Modify Downloaded Inquiries

✓ 20376,516884855,91116923350372803876607057259059629820

✓ 20375,516884855,39371921580965416229552994921770586476

Find the Intersection

✓ 20375,516884855,parkinson,

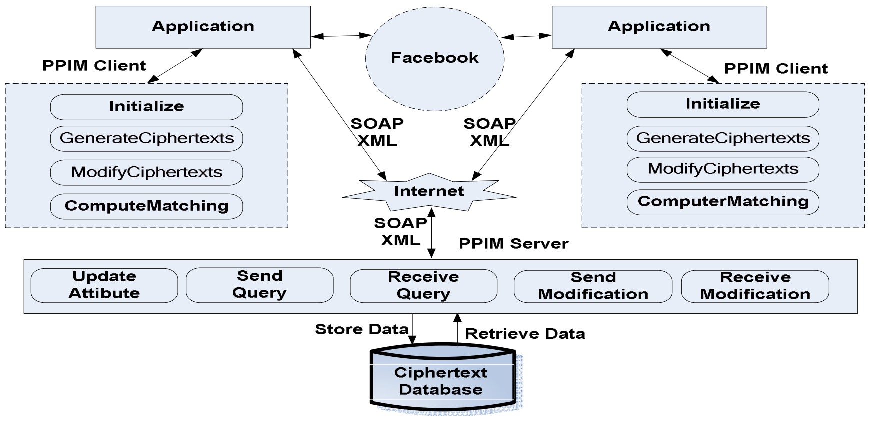


Fig. 12. The PatientMatch prototype. (a) The client tool (b) PPIM architecture.

In the application, Alice can try to locate peers based on the health conditions (left up corner of the GUI), discover potential matching candidates from SNS (next to the attribute selection on GUI), encrypt the attributes (middle left box of GUI), and upload the inquiries to PPIM server. Then, she needs to wait for matching candidates to respond on the submitted inquiries. On the right upper corner, two boxes with green labels represent matching invitations and their inquiries stored on the PPIM server. Alice can download inquiries of these invitations, and modify these inquiries using the modification function *F*. Then, she can upload them back to the PPIM server. On the right lower corner, Alice can download all her submitted inquires that have been modified by her matching candidates. She can then choose from the list of downloaded, modified inquiries to perform the matching to get the final outcomes.

We used MIRACL [86] as the crypto library. We chose G_1 as an additive group of points on ς: $y^2 = x^3 + x$, with prime order $q = 2^{159} + 2^{17} + 1$. All our parameters obtained from [86] deliver protection strength comparable to 1024-bit RSA. We chose *e* as Tate pairing [85], and the SHA-1 [87] as H_2.

We tested the running time of the PPIM software on different machines. First, we conducted a few experiments on an Intel Pentium-4 2-GHz processor with 256-Mbyte RAM and Windows XP. First we test the running time of SH. Given that other operations in SH like hash function takes less than 1 *ms,* we used the computation of Tate pairing [85] to represent the SH processing time. Basically two pairing computations that represent one successful SH execution need approximately 130 *ms*, which is efficient enough for practical use.

In the second experiment we measured the running time of one PSI transaction executed between two PPIM client tools for Alice and Bob on the same machine to minimize the communication delays. All messages are essentially encrypted elliptic curve points, one point per attribute, and the message size for each point is roughly 1K bytes. All the reported results are based the average of 100 runs of each experiment. The running times, including the (very small) communication delays, for one round of PSI matching for the 2*n* attributes between two users, where *n* = 10, 100, 1000, and 10000, is given in Table 7. When the client tool of Alice receives the modified attributes from that of Bob, it first fills up a comparison table after each modified attribute is multiple by its private key $PRI_{a.}$ Then, each entry in this table is directly compared against every entry of the inquiry sent by Bob. When *n* = 10 and 100, the total matching time is not increased much because the majority of time was spent on creation of the comparison table. When the number of attributes is large enough, the table creation time becomes less significant, and the comparison time becomes increasingly dominating. This leads to fairly linear increase of the overall PSI running time when *n* increased from 10 to 10k.

Table 7. Average running time of the ECC-based PSI protocol.

Number of Attributes	20	200	2k	20k
Time (sec)	6.266	7.750	22.953	175.436

Our experiments show that the computing cost is determined by the number of attributes that need to be matched. That is, when Alice uses 10 attributes to perform matching with other 1000 users, each of them also has 10 attributes; the total computing time is about the same as that Alice performs matching with one user who has 10k attributes for matching. In another setting, we fixed the number of attributes at n =10 for Alice, and the number of attributes for Bob at m, and the result is given in Table 8, with very similar performance characteristics as the first experiment: when n is small (large), the comparison table creation (matching) time dominates the other.

Table 8. Average running time of the PSI protocol.

(n, m)	10:10	10 : 100	10 : 1k	10 : 10k
Time (sec)	6.266	6.985	14.594	90.891

We also measured the matching time in a 100M bps local network environment, in which two PPIM clients run on two different machines M_A and M_B, and the PPIM server runs on machine M_C. Each request is fixed at the size of 10 attributes, and the running time for different parts of operations is itemed below. M_A generates the specified number of requests N_R in step S_1. It signals M_C about the new request in step S_2. M_C informs of M_B about the new matching request, in step S_3. Then, the request is sent from M_A, through M_C, to M_B, in step S_4. M_B modifies the received inquiry in step S_5, and then directly returns it to M_C. M_C first notifies M_A of the completion of the modification in step S_6, and then sends it the modified request. The cost of sending the modified inquiry from M_B, through M_C, to M_A, excluding that of S_6, is counted as step S_7. Finally, M_A

performs the final matching step in step S_8. The running time measurements of S_1 to S_8 are summarized in Table 9. The results suggest that the average cost T_A for handling one request is about a fraction of a second (0.12-0.13 sec), and the PPIM server is not considered a performance bottleneck.

Table 9 Running time (rounded) between three machines (unit: sec).

N_R	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	T_A
1	0.034	0.034	0.003	0.04	0.003	0.006	0.04	0.003	0.17
10	0.34	0.05	0.005	0.4	0.02	0.007	0.4	0.024	0.12
100	3.4	0.05	0.02	4.9	0.2	0.008	4.3	0.24	0.13
1k	34.5	0.05	0.1	45	2	0.045	44.3	2.45	0.128

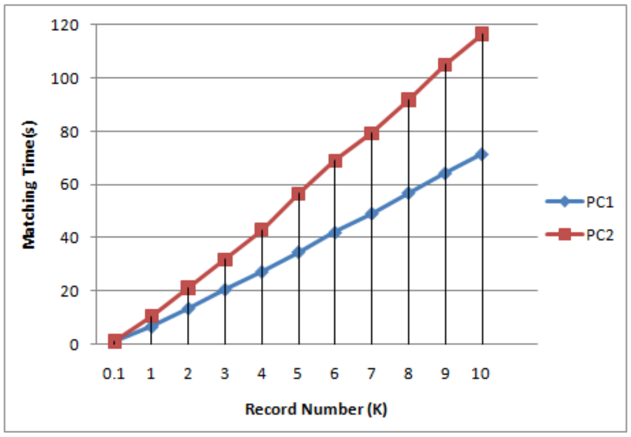
APPENDIX C

EVALUATION OF PAC

We have implemented PAC client tool using the crypto library of MIRACL [86] to test functionality and computing costs of the proposed algorithms. We chose G_1 as an additive group of points on E: $y^2 = x^3 + x$, with the prime order $q = 2^{159} + 2^{17} + 1$, and SHA-1 is selected [87] as H_1. We used the PAC client tool to correlate alerts generated from Snort, Autograph and honeynet traces for functionality testing.

The primary computing costs include the time required to perform matching, and the bandwidth for transmission of messages. We installed two copies of PAC client tools on two locally connected PC machines PC1 and PC2, (Intel(R) Xeon(TM) CPU 2.40GHz, RAM 2.00GB and Intel Core 2 Duo CPU 2.00GHz, 2.00GB RAM). For computing cost testing, we increased the number of attribute records (IP addresses extracted from a real honeynet trace) from 100 to 10000 on each machine to perform matching using Protocol II. Each experiment is repeated for twenty times to get the average computing costs, and the results are presented in Fig. 13 (a). The computing cost grows linearly with the number of attribute records: about one second for matching of 100 attribute records and less than 120 seconds for matching of 10000 attribute records for both two machines. The bandwidth requirement also grows linearly with the number of attribute records (see Fig. 13 (b)) based on the measurements made by Wireshark on PC1 [103].

We performed two experiments to derive the LCS between two inputs strings, whose lengths were set at 100 and 1000, respectively. For each input set, we further adjusted lengths of LCS, denoted as |LCS|, from 0 (matching of *l*-grams alone), to half input length, and to the full input length. We chose $l = 5$ for all experiments. The running times for these experiments are given in Table 10. The major costs of Protocol VI are associated with protection of locations of matched *l*-grams, i.e., randomization and shuffling, and the matching caused by the inserted faked points. We can see that the computing cost is linear to the input string length when the percentage p is fixed.



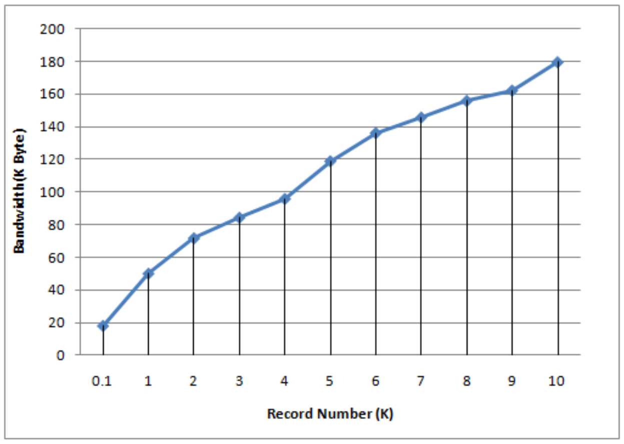


Fig. 13. (a) Processing time, and (b) Bandwidth for Protocol II of PAC.

Table 10. Average running time of the PPLCS protocol.

Input lengths: 100	Time (s)	Input lengths: 100	Time (s)
LCS length = 0	1.02	LCS length = 0	10.11
LCS length = 50	12.46	LCS length = 500	127.88
LCS length = 100	22.05	LCS length = 1000	221.94

We tested discovery of attack patterns using a honeynet packet trace captured in a ten-day period in 2007. This original trace contains successful host infections from external attackers, and a short period of infection propagation. To simulate correlation of source-destination IP addresses across two different networks, we divided the original trace by time and then by IP address to generate 4 traces. Trace 1(2) covers the first (second) period of 5 days that contains 131k (21k) packets for 6177(5246) distinct IP addresses. Trace 3(4) covers subnet I (II) for the 10 day period, and it contains 69k (83k) packets for 5215 (6275) distinct IP addresses. For common attacker/victim correlation, we extracted IP addresses from each trace, and then perform PAC based correlation between trace 1 and trace 2, and then between trace 3 and trace 4, respectively. We found 10 common attackers and 12 common victims between trace 1 and trace 2, and 58 common attackers and 70 common victims between trace 3 and trace 4, respectively. The example in Fig. 14 shows that the common attacker (right hand side) was discovered from the two traces (left hand side), which have different packet counts for the identified IP. Here the term *scale* records the order of magnitude of a particular IP being correlated and the first two bytes of the IP addresses were replaced by symbols to protect network identities.

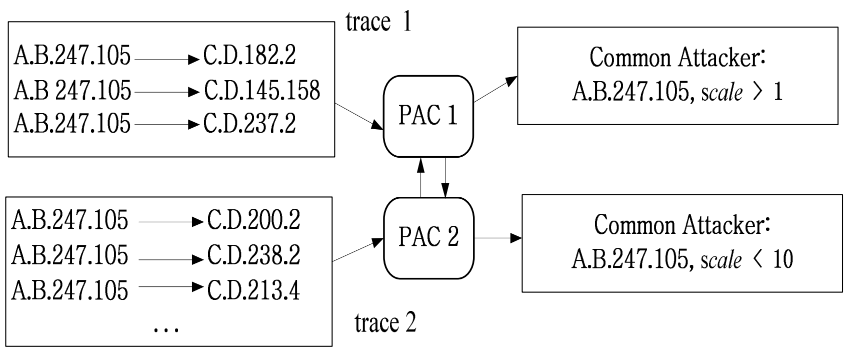


Fig. 14. An example of common attacker discovered by PAC.

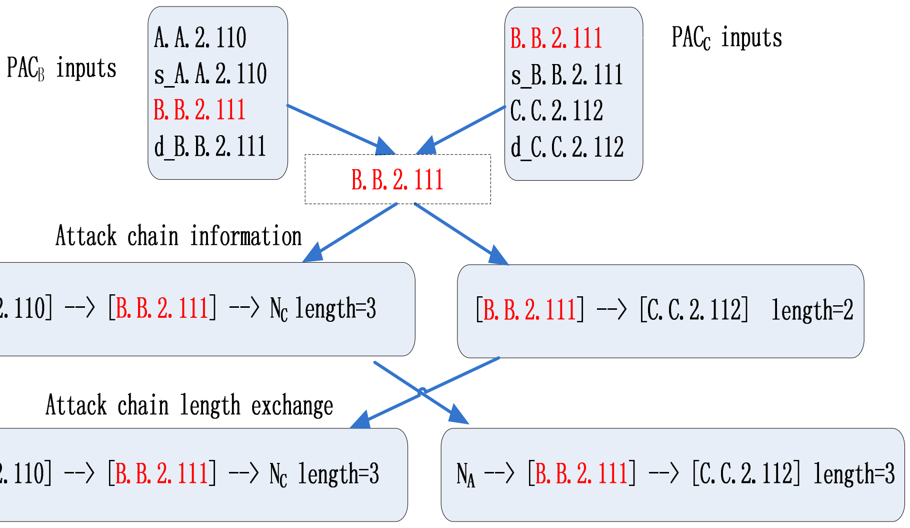
An attack chain often span across multiple networks. It requires cooperation of all involved networks to reconstruct the connection topology. The PAC client tool allows any two adjacent points on the chain to confirm their existence, i.e., the *neighborhood* relationship of two networks on an unknown attack chain. One option for the next level correlation is based on an attack chain length calculation technique. That is, two confirmed neighbors can exchange the length information (about the unknown attack chain) that they already received from other neighbors, and then produce their own latest lengths. When a network finds that the length of an attack chain exceeds a threshold, it can spread the information for follow up actions.

Without loss of generality, we used the attack chain of Agobot infection to illustrate the key steps involved in the process. In the experiment, five virtual hosts (H_A-H_E), each of runs an un-patched Windows 2000 with *DCOM* vulnerability, were placed in five different networks N_A, N_B, N_C, N_D, N_E. Snort configured with *DCOM-Port-Scanning* rules was also installed on these networks to monitor inbound traffic. A simulated botmaster instructed the Agobot client on H_A to infect four other hosts one by one. Snort$_B$ running on N_B detects the intrusion from H_A and H_B is its victim. Similarly, $Snort_C$, $Snort_D$, and $Snort_E$ also generated their local alerts. Administrators of N_B and N_C have agreed to use their PACs to correlate their victim IP addresses vs. attack source IP addresses.

When PAC_B and PAC_C respectively have the victim IP addresses (of N_B) and attack source addresses (of N_C) as their inputs, both PAC_B and PAC_C conclude that H_B is matched. As a result, N_B (N_C) knows the existence of an attack chain $H_A \rightarrow H_B \rightarrow N_C$ $(H_B \rightarrow H_C)$, but N_B does not know which host on N_C has been attacked and N_C does not know H_B was also attacked by H_A. After a length exchange, Both N_C and N_B know that the length of the attack chain is 3, i.e., $H_A \rightarrow H_B \rightarrow N_C$ $(N_A \rightarrow H_B \rightarrow H_C)$. Following a similar argument, PAC_C and PAC_D can conclude that they are respectively on an attack chain fragment of $N_A \rightarrow H_B \rightarrow H_C \rightarrow N_D$ *and* $H_C \rightarrow H_D$ after they complete their alert correlation. After length exchange, N_C and N_D know that they are on an attack chain of length 4. Neighboring networks can inform each other about an attack chain whenever the length discovered by the correlation process exceeds a threshold.

Snapshots from the experiments are illustrated in Fig. 15. Each IP address with its discovered appearing times were fed to PAC for matching. For example, the entry *B.B.2.111* (in red color) indicates the discovered IP address of H_B, and below it *d_B.B.2.111* indicates that it is a destination IP. After their matching, N_B and N_C can both see that they are the neighborhood of each other: N_B gets *A.A.2.110*$(H_A) \rightarrow$*B.B.2.111*(H_B) $\rightarrow N_C$ (*B.B.2.111* attacks an unknown IP in *C*'s network). On the other hand, N_C gets

B.B.2.111(H_B)→ C.C.2.112*(H_C)*. After they exchange their length information, both networks know that the existence of an attack chain of three. The correlation process can be indefinitely expanded between any two networks. The length information can also be generalized to include more information, e.g., branches, so that an arbitrary attack chain topology can be derived based on our scheme.


PAC_B inputs
A.A.2.110
s_A.A.2.110
B.B.2.111
d_B.B.2.111
B.B.2.111
s_B.B.2.111
C.C.2.112
d_C.C.2.112
PAC_C inputs
B.B.2.111
Attack chain information
[A.A.2.110] --> [B.B.2.111] --> N_C length=3
[B.B.2.111] --> [C.C.2.112] length=2
Attack chain length exchange
[A.A.2.110] --> [B.B.2.111] --> N_C length=3
N_A --> [B.B.2.111] --> [C.C.2.112] length=3
Fig. 15. Illustration on discovery of the neighborhood on an infection chain.

PAC can be used for cross-network correlation to derive more global signatures by exclusion of as much local information as possible in generation of their LCS. As discussed earlier, other shorter common substrings can also be iteratively generated using Protocol VI. To evaluate Protocol VI in PAC, we used local signatures generated by Autograph run on different networks as the inputs to generate their LCS. Four types of traffic were used in the example. The type I traffic is consisted of CodeRed I and CodeRed II worms. The other three types of traffic (type II, III and IV) are plain text, HTTP, and some binary executable files of regular applications. The testbed setup for this experiment is illustrated in Fig. 16. In the first phase of the experiment, we let hosts *E* & *F* scan ports on host *B* in network 1 and host *C* in network 2, respectively. Two Autographs deployed at network 1 and network 2 detected the port scanning and then added *E* and *F* to their suspicious IP lists for network 1 and network 2, respectively. All subsequent packets sent from *E* and *F* to the scanned ports of *B* and *C* were classified as abnormal traffic. In phase 2, hosts *E* and *F* began to transmit mixed traffic of types I-IV to the two domains. CodeRed I (II) traffic was included in the traffic. Using signatures created by Autograph in the two different networks as inputs, we generated the LCS for CodeRed I and CodeRed II and removed the other types of traffic as unmatched (local) contents.

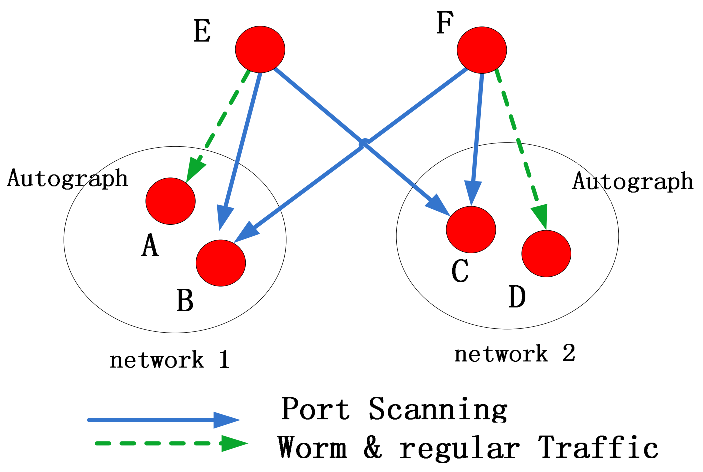


Fig. 16. Attack patterns of CodeRed I & CodeRed II in the testbed.

In a similar experiment, we first used Autograph to extract local signatures from local alert trace 1 and trace 2 separately. Most signatures came from attempts to exploit vulnerable services such as HTTP (port 80), Netbios-ssn (port 139), Microsoft-ds (port

445). Then, we used PAC to determine the common signature between these local signatures. Experimental results showed that Autograph totally found seven (four) local signatures from trace 1 (2). We were able to generate the common signature correlated from local ones for a malware instance that successfully exploited Microsoft-ds service in both traces.

VITA

Name:	Pu Duan
Address:	Room 503, H.R. Bright Building, Department of Computer Science and Engineering, Texas A&M University, TAMU 3112, College Station, TX 77843-3112
Email Address:	duanpu1979@tamu.edu
Education:	B.S., Information Engineering, Xi'an Jiaotong University, 2001
	Ph.D., Computer Science, Texas A&M University, 2011