US 20040030894A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2004/0030894 A1**

Labrou et al. (43) **Pub. Date: Feb. 12, 2004**

(54) **SECURITY FRAMEWORK AND PROTOCOL FOR UNIVERSAL PERVASIVE TRANSACTIONS**

(75) Inventors: **Yannis Labrou**, Baltimore, MD (US); **Lusheng Ji**, Silver Spring, MD (US); **Jonathan Russell Agre**, Brinklow, MD (US)

Correspondence Address:
STAAS & HALSEY LLP
SUITE 700
1201 NEW YORK AVENUE, N.W.
WASHINGTON, DC 20005 (US)

(73) Assignee: **FUJITSU LIMITED**, Kawasaki (JP)

(21) Appl. No.: **10/458,205**

(22) Filed: **Jun. 11, 2003**

Related U.S. Application Data

(60) Provisional application No. 60/401,807, filed on Aug. 8, 2002.

Publication Classification

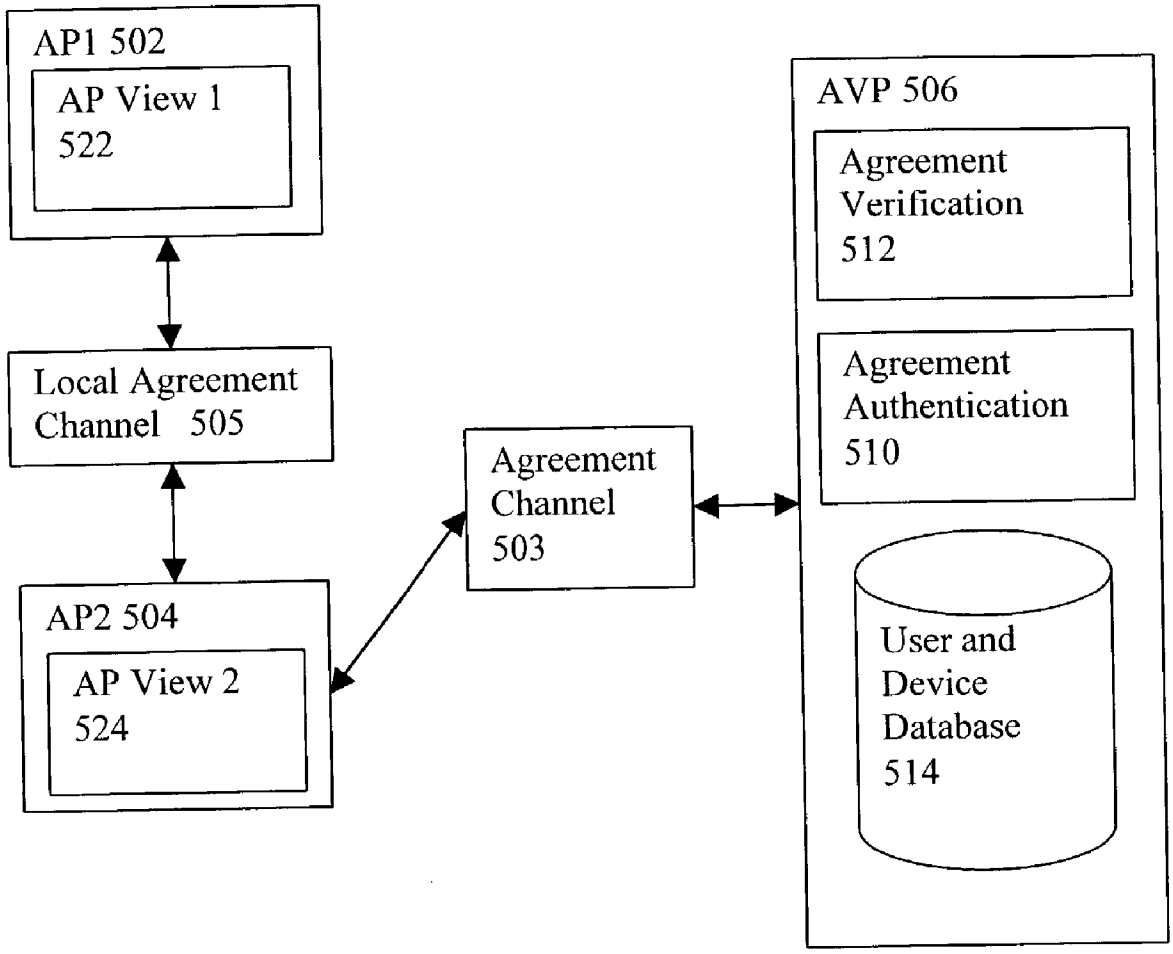
(51) **Int. Cl.**[7] **H04L 9/00**

(52) **U.S. Cl.** **713/168**

(57) **ABSTRACT**

A computer system, a method of a computer system and a computer-readable medium securely transmit and verify a multiparty agreement. The method, the computer system, and the computer readable medium include developing and transmitting views of the multi-party agreement by each party to a separate verification party. The verification party authenticates the participants and determines whether the views of the agreement are mutually consistent, and notifies the partys of the results of the comparison.

500



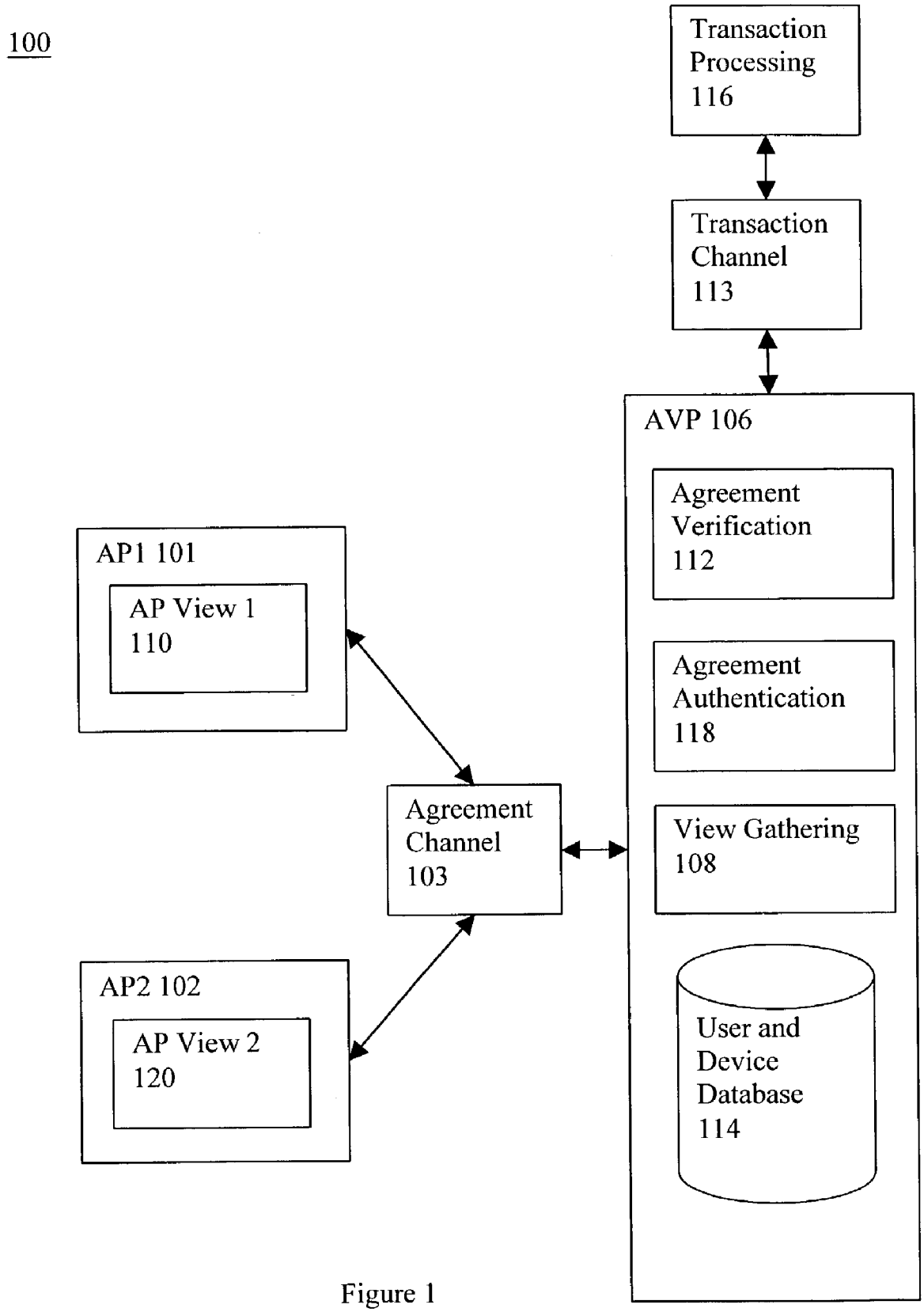

100
Transaction Processing 116
Transaction Channel 113
AVP 106
Agreement Verification 112
Agreement Authentication 118
View Gathering 108
User and Device Database 114
AP1 101
AP View 1 110
Agreement Channel 103
AP2 102
AP View 2 120

Figure 1

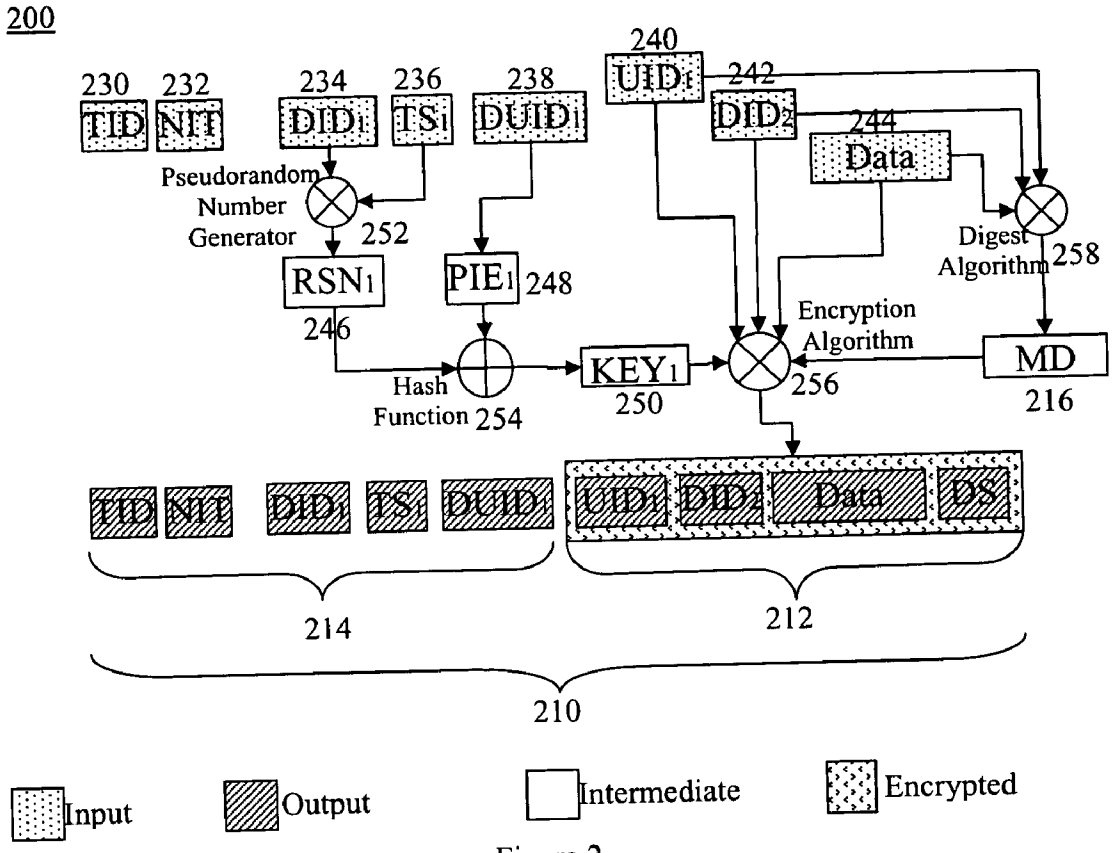

200
230
TID
232
NIT
234
DID1
236
TS1
238
DUID1
240
UID1
242
DID2
244
Data
Pseudorandom Number Generator
252
RSN1
246
PIE1
248
Hash Function
254
KEY1
250
Encryption Algorithm
256
Digest Algorithm
258
MD
216
TID
NIT
DID1
TS1
DUID1
UID1
DID2
Data
DS
214
212
210
Input
Output
Intermediate
Encrypted

Figure 2

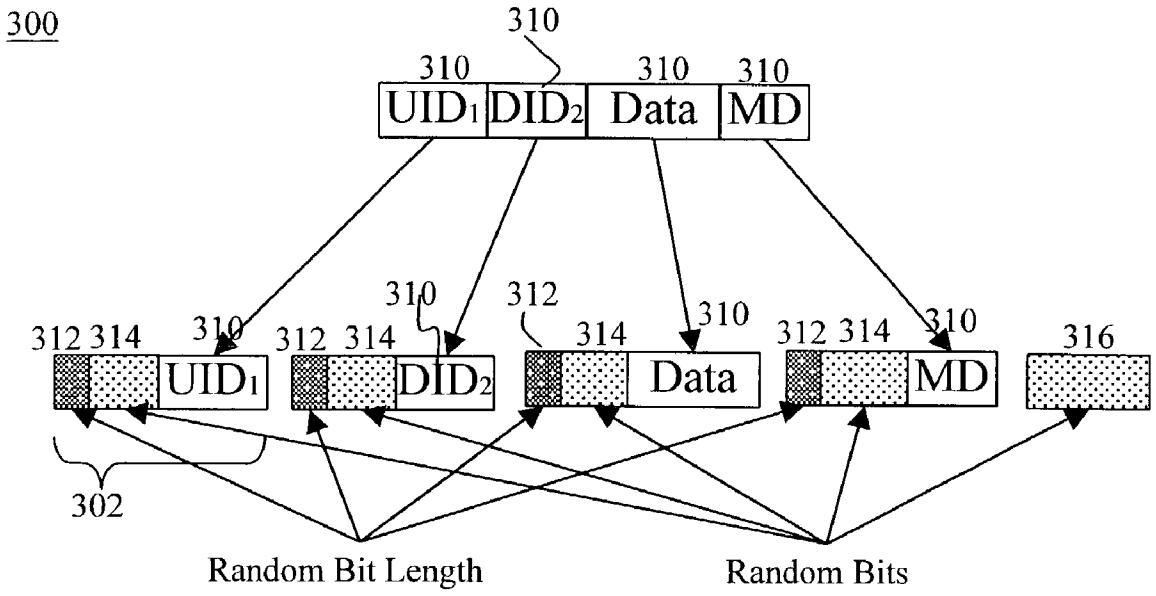

300
310
310
310
310
UID1
DID2
Data
MD
312 314
310
312 314
310
312
314
310
312 314
310
316
UID1
DID2
Data
MD
302
Random Bit Length
Random Bits

Figure 3

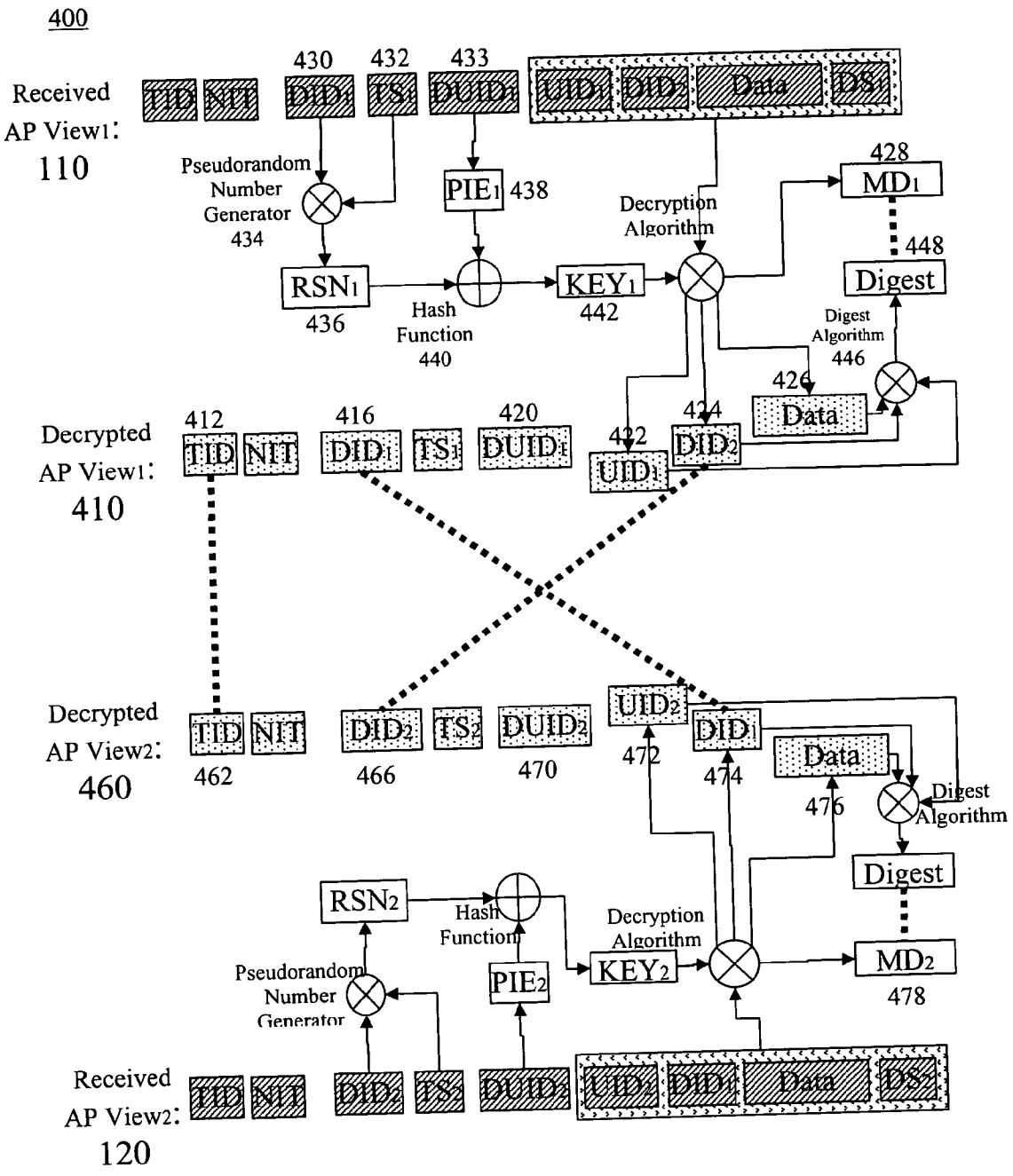

400
Received AP View1: 110
TID
NIT
DID1
TS1
DUID1
UID1
DID2
Data
DS1
430
432
433
Pseudorandom Number Generator 434
PIE1 438
Decryption Algorithm
428
MD1
448
Digest
RSN1 436
Hash Function 440
KEY1 442
Digest Algorithm 446
426
Data
Decrypted AP View1: 410
412
416
420
422
424
DID2
Decrypted AP View2: 460
462
466
470
472
474
476
Digest Algorithm
Digest
RSN2
Hash Function
Decryption Algorithm
Pseudorandom Number Generator
PIE2
KEY2
MD2
478
Received AP View2: 120

Figure 4

500

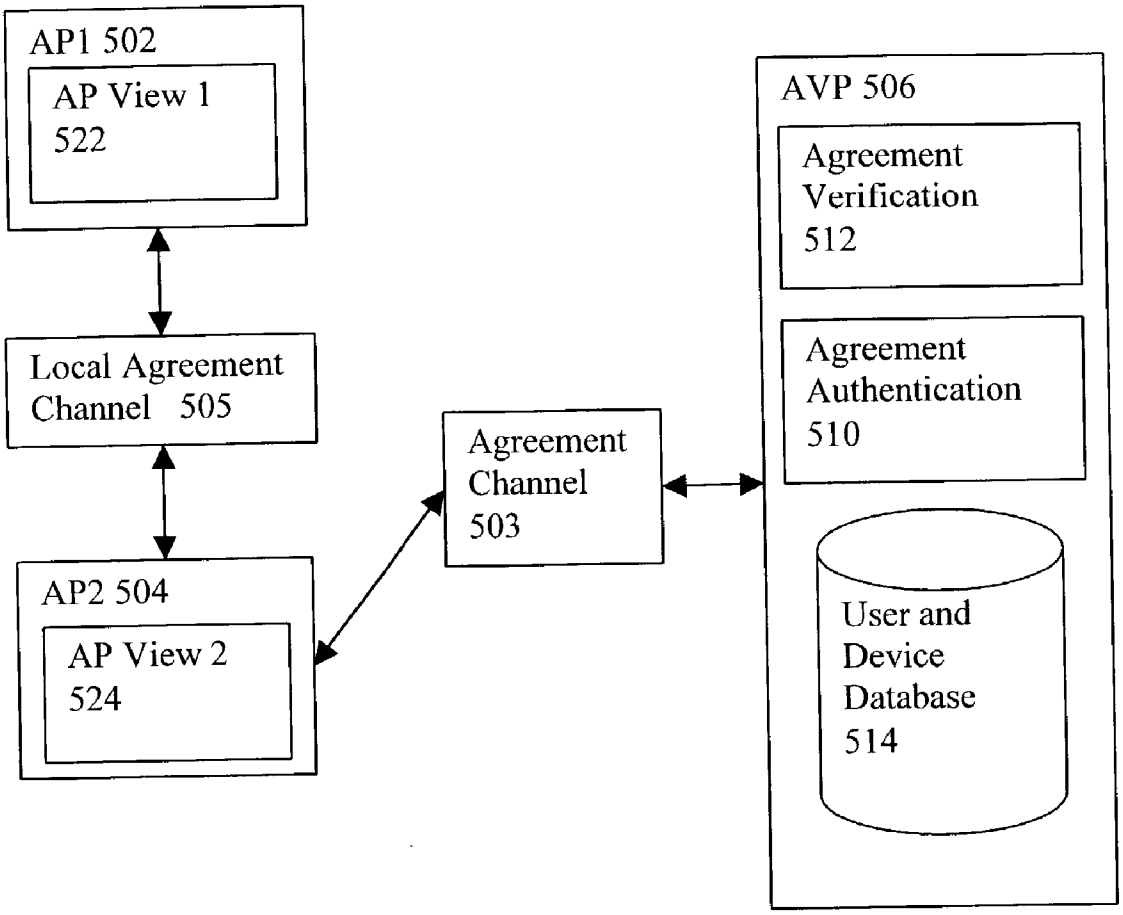

AP1 502
AP View 1
522
Local Agreement
Channel 505
AP2 504
AP View 2
524
Agreement
Channel
503
AVP 506
Agreement
Verification
512
Agreement
Authentication
510
User and
Device
Database
514

Figure 5

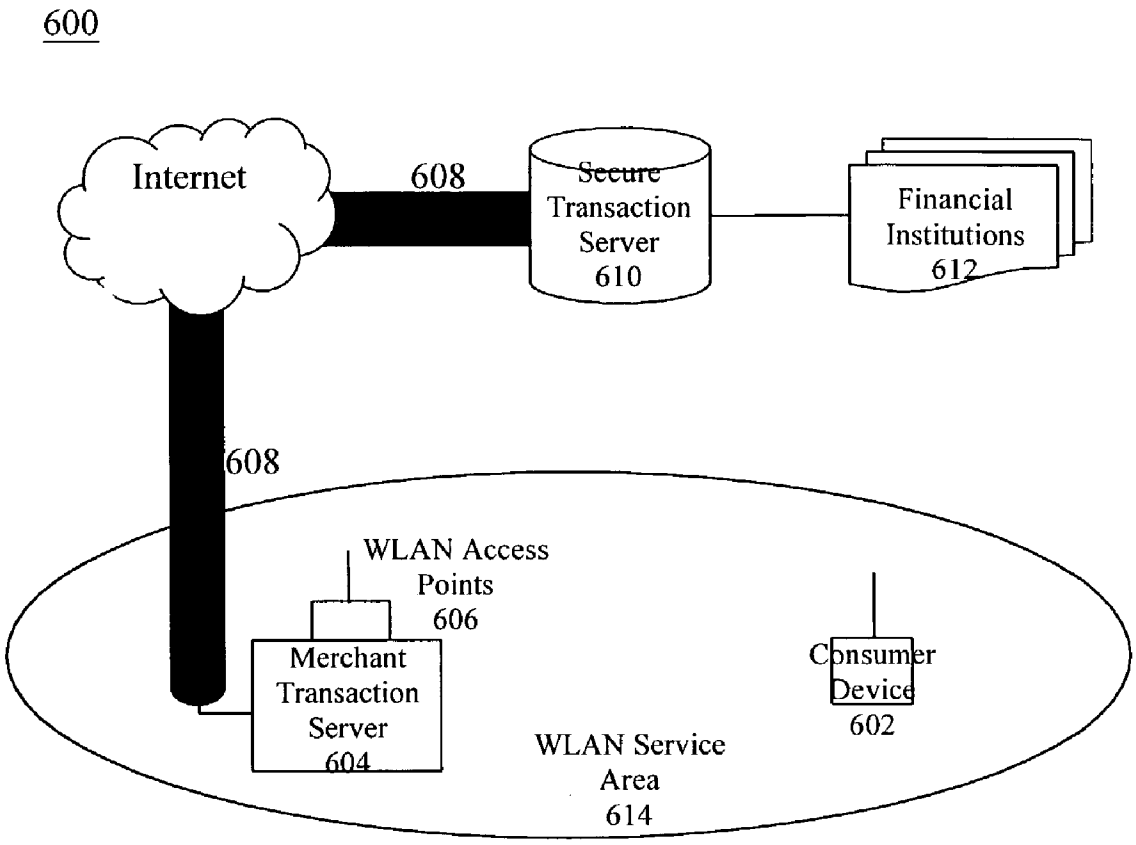

600
Internet
608
Secure Transaction Server 610
Financial Institutions 612
608
WLAN Access Points 606
Merchant Transaction Server 604
Consumer Device 602
WLAN Service Area 614

Figure 6

700

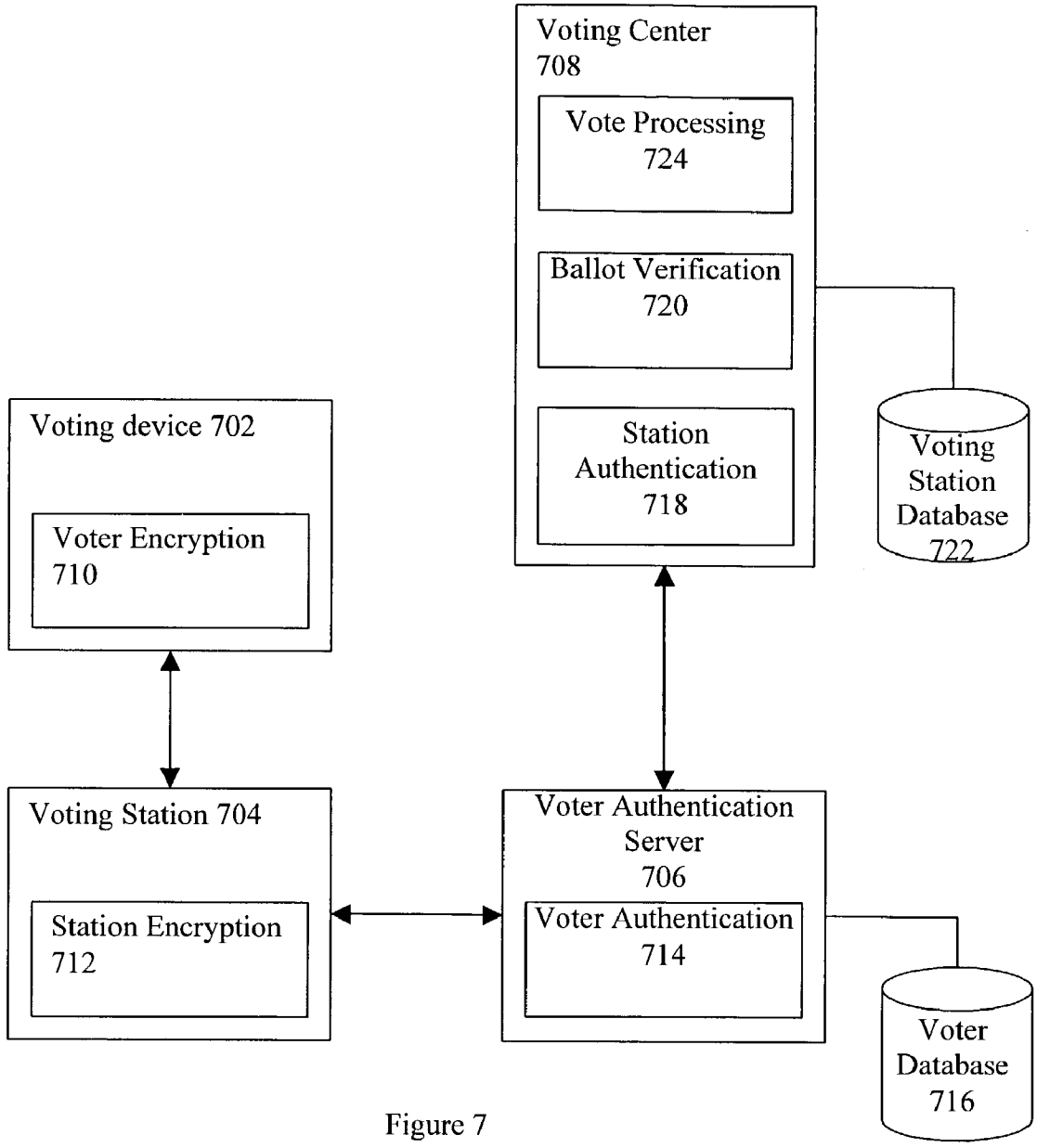

Voting Center
708
Vote Processing
724
Ballot Verification
720
Station
Authentication
718
Voting
Station
Database
722
Voting device 702
Voter Encryption
710
Voting Station 704
Station Encryption
712
Voter Authentication
Server
706
Voter Authentication
714
Voter
Database
716

Figure 7

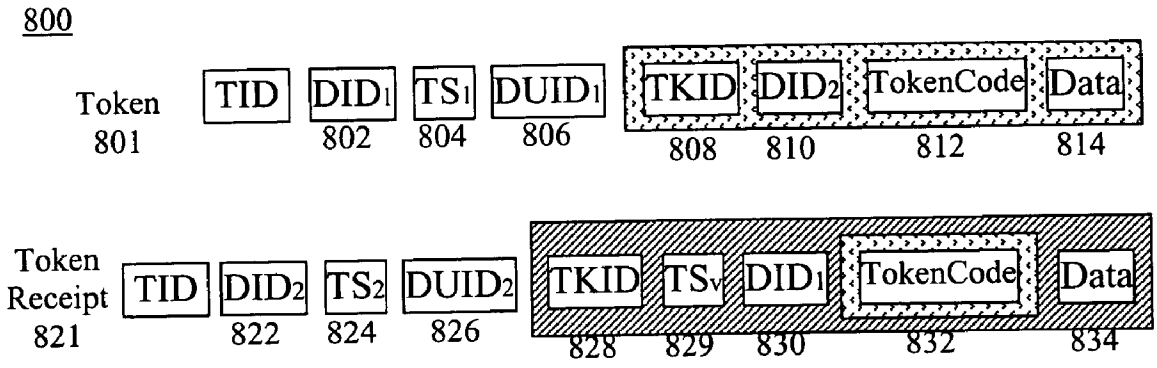

800
Token
801
TID
DID1
802
TS1
804
DUID1
806
TKID
808
DID2
810
TokenCode
812
Data
814
Token
Receipt
821
TID
DID2
822
TS2
824
DUID2
826
TKID
828
TSv
829
DID1
830
TokenCode
832
Data
834

Figure 8

SECURITY FRAMEWORK AND PROTOCOL FOR UNIVERSAL PERVASIVE TRANSACTIONS

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application is related to, and claims the benefit of priority to, Provisional Application U.S. Serial No. 60/401,807, Attorney Docket No. 1634.1002P, entitled METHODS AND APPARATUSES FOR SECURE MULTI-PARTY FINANCIAL TRANSACTIONS (A UNIVERSAL PERVASIVE TRANSACTION FRAMEWORK), by Yannis Labrou, Lusheng Ji, and Jonathan Agre, filed Aug. 8, 2002 in the U.S. Patent and Trademark Office, the contents of which are incorporated herein by reference.

BACKGROUND OF THE INVENTION

[0002] 1. Field of the Invention

[0003] The present invention is related to computer communications, and, more particularly, to secure communications between remote computers.

[0004] 2. Description of the Related Art

[0005] Symmetric cryptographic schemes (or algorithms), in which encryption and decryption use the same key, are well known in the art and have several desirable characteristics such as ease of key management and lower computational requirements as compared to asymmetric cryptographic schemes.

[0006] Many current security mechanisms employ asymmetric cryptographic schemes, such as the public key systems with their associated Public Key Infrastructure (PKI) systems and are known in the art. However, the PKI (Public Key Infrastructure) system of the related art includes specific costs associated with creating and maintaining this infrastructure. Examples of these costs include key distribution, management and storage.

[0007] The asymmetric encryption/decryption algorithms used by the PKI systems involve relatively complex and time-consuming computations. Hence they are not well suited for economical and compact mobile computing devices on which only limited computing resources and battery power are available.

[0008] Symmetric algorithms consume substantially less computing power than asymmetric encryptions and decryptions. Communicating parties in symmetric cryptographic systems typically share the same key, which is then used by them as a parameter to encrypt and decrypt the message data.

SUMMARY OF THE INVENTION

[0009] An aspect of the present invention is to require less computation on client devices when applying encryption/decryption to plain text data as well as less effort managing the encryption/decryption keys than PKI systems known in the art.

[0010] Another aspect of the present invention is to provide a novel, shared algorithm to devise a key without sharing the entire key itself, as compared with other typical secure communication systems based on symmetric cryptography.

[0011] A further aspect of the present invention is to provide a method which protects the integrity of a group communication, but does not rely on a shared secret key among the group members. This method not only prevents any party not belonging to the group from participating in group communication; it also detects the absence of any communication group member.

[0012] Still another aspect of the Secure Agreement Submission protocol (SAS) of the present invention is that a third party can not attempt to guess the key or parameters of the key derivation scheme through the generation of SAS messages without a high likelihood of detection by the verification party.

[0013] The present invention relates to a method of a third party (verification party) verifying an agreement between two distrusting parties (agreement parties) in an insecure communication environment. The present invention extends to a multi-party agreement method, where a verification party verifying an agreement among multiple (more than two) distrusting agreement parties in an insecure communication environment.

[0014] The present invention is a computationally light-weight protocol carrying agreement data and other sensitive messages between distrusting agreement parties and a verification party in an insecure communication environment so that the agreement data is protected during the transmission and the agreement data can be shown to be consistent. The protocol of the present invention satisfies security properties such as privacy, authentication, user anonymity, non-replayability and non-repudiation.

[0015] The present invention defines a Secure Agreement Submission (SAS) protocol that is designed for use in unreliable communication environments, such as wireless networks. The SAS of the present invention enables multiple parties to an agreement to submit the agreement information to an independent verification party in a secure fashion over these unreliable communication channels. In addition, the SAS of the present invention provides a mechanism and procedures comparing and verifying the agreement information and notifying the participants of the results, also in a secure fashion. As is disclosed herein below, the present invention is ideally suited for many types of transactions such as purchasing goods, wireless voting, virtual token collection and many others.

[0016] The SAS of the present invention includes a cryptographic scheme based on a family of symmetric cryptography algorithms, in which encryption and decryption use the same shared key. The SAS of the present invention includes a novel key derivation and generation scheme that can be used with many symmetric cryptographic schemes and results in several new, desirable properties for the protocol, such as a high degree of security in a non-secure communication environment (such as a wireless channel), low computational complexity and no need for a user to store or transmit keys, or other personal identification data pertaining to the attempted agreement, such as username, account data, etc.

[0017] The key generation scheme of the present invention uses a mobile computing device capable of communication. The mobile computing device executes the protocol and accepts input from a user. Such devices can be special

purpose devices or readily available computing platforms such as Personal Digital Assistants or programmable cellular or mobile telephones.

[0018] The key derivation algorithm of the present invention combines information about the mobile computing device with information about the user of the device. The algorithm also combines information that is stored digitally by the device and the shared secret information that is input by the user. Such a combination ensures with high likelihood that only the intended parties are able to decrypt and thus access the communicated data. If a device is lost or stolen, it can not be used without the specific user input information, which itself is not stored on the device. The deterministic key derivation algorithm may be generally known. The set of stored parameters is preferably known only to the device and the verification party, but if generally known are not sufficient to determine the key, without knowledge of the shared secret value. The secret value, or the stored parameters, or the key are never transmitted in a message. What is transmitted is a message parts of which are encrypted with a key that is derived from the stored parameters and the shared secret information that is input by the user.

[0019] These together with other aspects and advantages which will be subsequently apparent, reside in the details of construction and operation as more fully hereinafter described and claimed, reference being had to the accompanying drawings forming a part hereof, wherein like numerals refer to like parts throughout.

BRIEF DESCRIPTION OF THE DRAWINGS

[0020] **FIG. 1** is an example of a computer system in which the security agreement submission protocol (SAS) view of the present invention is implemented.

[0021] **FIG. 2** shows a method of encrypting a security agreement submission protocol (SAS) view of the present invention.

[0022] **FIG. 3** shows a method of decrypting a security agreement submission protocol (SAS) view of the present invention and how the cross reference fields are matched.

[0023] **FIG. 4** is another example of a computer system in which the security agreement submission protocol (SAS) view of the present invention is implemented.

[0024] **FIG. 5** illustrates how random bit padding is applied to encrypted data fields.

[0025] **FIG. 6** shows an example application of the present invention in purchasing of goods and services.

[0026] **FIG. 7** shows a different example application of the present invention applied to electronic voting.

[0027] **FIG. 8** illustrates how the present invention can be used to generate 3rd-party verifiable tokens.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0028] An overview of the present invention is now presented.

[0029] An agreement, with respect to an application, is a general statement between parties for which a verification procedure can be executed to provide confirmation that the parties have a common understanding of the statement, within the context of that application. For example, a financial transaction agreement could be that "Party A will pay Party B $X for item Y." An agreement statement is represented by agreement data, the contents of which are not defined by the invention but by the needs of the application.

[0030] The method of the present invention includes a new protocol, referred to as the Security Agreement Submission (SAS) protocol, to accomplish the agreement verification. An aspect of the present invention is an SAS encryption (SASE) mechanism that provides many security properties in an insecure communication environment. The SASE is used to encrypt and decrypt all messages that are part of the SAS. The SASE mechanism is implemented by each of the agreement parties and the verification party.

[0031] The present invention achieves the following desirable security properties:

[0032] Authentication of agreement parties: The identities of the involved agreement parties can be determined to be who they claim they are, to a high degree of likelihood by the verification party, based on the fact that a SASE coded message sent by an agreement party can be decrypted and understood by the verification party, using a decryption method with a key that is specific to the sender and only known to the verification party and the specific agreement party.

[0033] Authentication of verification party: The identity of the verification party can be determined to be who it claims it is, to a high degree of likelihood by each individual agreement party, based on the fact that a SASE coded message sent by the verification party for a particular agreement party can be decrypted and understood only by that agreement party using a decryption method with a key specific to the agreement party and only known to the agreement party and the verification party;

[0034] Anonymity: The agreement parties may remain anonymous to each other, if desired in an application through the use of the SASE method.

[0035] Privacy of Agreement: The agreement data sent between the agreement parties and the verification party is protected by SASE so that, if intercepted, no party other than the intended receiver is able to decrypt and read the data. Similarly, response messages from the verification party to the agreement parties are protected.

[0036] Tamper-resistance: The agreement data sent between the agreement parties and the verification party is protected through the use of an encryption signature so that no party can alter the data sent by other parties without a high degree of detection.

[0037] Non-replayable: Agreement data sent between the agreement parties and the verification party (if intercepted) is protected by an encryption mechanism that incorporates the value of the time when the agreement transaction occurs, and such a timestamp is also included in each message and recorded by the verification party. Thus, no party can replay the agreement data to forge a new agreement because each key is associated with a specific timestamp which is recorded by the verification party in a message log.

[0038] Non-repudiation: An agreement party can not later claim that they did not generate an agreement message that

has been verified by the verification party except under certain specific conditions that are highly unlikely. These security breeches include the case, where all the secret parameters (the device-specific stored parameters and the shared secret which is input by the user of the device) have been divulged or discovered and the mobile-computing device has been used without the consent of that agreement party. It is also possible for the verification party to generate a false agreement, but it would involve the collusion of the verification party and the other parties to the agreement, which is also highly unlikely. In addition, the verification party will keep records that record the sequence of SAS message exchanges involved in each transaction.

[0039] Agreement Group Authentication: The present invention ensures the integrity of the agreement party group (the group consisting of and only of the parties among which the agreement is conducted) so that no other party can pretend to be an agreement party or an agreement party can pretend not to be an agreement party. This is accomplished explicitly by a membership list and identity cross-referencing. It is also assumed that all participants in the agreement are a priori known to the verification party and able to be individually authenticated.

[0040] Agreement Verification: The agreement is verified to be consistent among the authenticated agreement parties through the use of redundant and cross-referencing information contained in the agreement data and the use of a verification procedure consisting of basic matching rules and specific matching rules that may depend on the application.

[0041] Computational Efficiency: The security mechanism of the present invention is based on private key (symmetric) cryptography that is more efficient than alternative methods.

[0042] Physical Security: The security mechanism can be implemented so that it is not necessary to store all of the necessary encryption information on the client mobile computing devices, thus making it easier to protect the secret information if the device is compromised. Specifically, the shared secret input by the user is not stored on the device. Also, when the device is used in a particular application context, user-identifying information is not stored on the device. For example, when the device is used for purchasing goods and service in physical retail stores, the name of the consumer, or the user's account information is not stored on the device.

[0043] Intrusion Detection: The security mechanism is centralized through the use of an independent verification party so that attempts to use the system by unauthorized users that rely on multiple access attempts are easily detected and handled accordingly.

[0044] With the above-mentioned aspects of the present invention, the present invention is ideal for being used as a vessel to carry financial transaction data between distrusting parties in an insecure communication environment. It is also well-suited for a system using low-cost user devices, which have limited computing resources.

[0045] The present invention is now explained with reference to FIGS. **1-8**.

[0046] Architecture

[0047] The overall architecture of a system **100** for agreement verification between two parties using the SAS of the present invention is shown in **FIG. 1**. The system **100** comprises two Agreement Parties, AP1 (**101**) and AP2 (**102**), an Agreement Communication Channel (**103**), the Authentication and Verification Party AVP (**106**), a Transaction Communication Channel (**113**) and Transaction Processing Component (**116**). The AVP **106** itself comprises four components, the View Gathering Module (**108**), the Agreement Authentication Module (**118**), the Agreement Verification Module (**112**), and the User and Device Database (**114**).

[0048] Referring now to **FIG. 1**, AP1 **101** generates agreement information in the form of AP View 1 (**110**) and AP2 **102** generates agreement information in the form of AP View 2 (**120**). The Transaction Processing Component **116** and its associated communication channel are not part of the present invention, but are included to further illustrate the application environment for the present invention. It is assumed that the Transaction Communication Channel **113** is a reliable and secure channel.

[0049] The present invention assumes that the Agreement Channel **103** is a reliable, although insecure, communication channel between the APs **101**,**102** and the AVP **106**. All messages that are part of the SAS protocol are encrypted/decrypted using the SASE. In addition, the AVP **106** is considered to be located in a secure facility, so that the sensitive information in the User and Device Database **114** is sufficiently protected.

[0050] The SAS agreement verification process of the present invention is described as the following six functions. More details of each function are provided in the later sections:

[0051] Function 1: Each Agreement Party (AP) **101** or **102** creates the AP View **110** or **120** including agreement data and additional parameters. Sensitive portions of the view **110** or **120** are encrypted using the SASE of the present invention. The AP View **110** or **120** is digitally signed by the AP **101** or **102**, respectively. An Agreement Message is created from the view **110** or **120** and then transmitted to the Authentication and Verification Party (AVP) **106** using the Agreement Communication Channel **103**.

[0052] Function 2: The AVP **106** receives the agreement messages from the APs **101** or **102** and delivers them to the View (or Agreement) Gathering Module **108**. The View Gathering Module **108** determines that this is a two-party agreement and when it has received two agreement messages (one from each party) for this particular agreement. The messages are then passed to the Authentication Module **118**.

[0053] Function 3: The Authentication Module **118** authenticates the agreement parties by using the SASE of the present invention to decrypt the agreement messages, and determines that the signed agreement copies are indeed signed by the involved APs **101** or **102**. This is done through the properties of the SASE scheme and using the information stored in the User and Device Database **114**. If authenticated, then the decrypted messages are passed to the Agreement Verification Module **112**. If the authentication fails, then the results are sent to the Agreement Parties **101** or **102** as indicated in Function 6.

[0054] Function 4: The Agreement Verification Module **112** executes a set of matching rules that check to determine whether the agreement data in each of the agreement mes-

sages **110** and **120** is consistent with each other. There are several matching rules that are always applied as well as an interface for application-specific rules. Together these matching rules are checked to verify that the agreement data included in all received copies of the agreement is consistent. Typically, in each agreement message there is reference to the other parties of the agreement and possibley a reference to a user identity that is not public information (for multiple users per device case). In addition, each application of the present invention can provide a plug-in function to verify that the application specific contents of the agreement received from the agreement parties agree with each other. For example, in a financial transaction, there is an agreed upon amount that can be matched among the parties. If there is no associated transaction processing, then the system proceeds to Function 6. Otherwise, Function 5 is then executed.

[**0055**] Function 5: In many applications, once the agreement details have been verified, it is desirable to perform some services based on the contents of the agreement. In this case, the decrypted agreement data is passed to the Transaction Processing Component **116** to execute these services using the Transaction Communications Channel **113**. The Transaction Processing Component **116** will typically create response messages for each agreement party following the processing of the transaction. The response messages are communicated back to the Agreement Verification Module **112** through the same channel.

[**0056**] Function 6: The Agreement Verification Module **112** creates a Response Message for the Agreement Parties **101** or **102** that includes the results of the verification. If there is a response from a Transaction Processing Component **116**, then this is also incorporated into Response Messages. The Agreement Verification Module **112** passes the response messages to the Agreement Authorization Module **118** that uses the SASE of the present invention to encrypt response messages for the Agreement Parties **101** or **102** and transmit the response messages to the agreement party **101** or **102** over the Agreement Communication Channel **103**.

[**0057**] The agreement method of the present invention is summarized herein above. However, in order to operate such a system **100** implementing the agreement method of the present invention, there are several additional functions that occur. Prior to joining an agreement, any AP **101**,**102** who wishes to use the verification service must be registered with the Authentication and Verification Party (AVP) **106**. The registration process results in a user account being created for the AP **101** or **102** at the AVP **106** and necessary information stored in the User and Device Database **114**. A registered AP is hence known as an AP User of the system.

[**0058**] Registered APs **101**,**102** are assumed to employ devices, called AP Devices or Client Devices. Each device is capable of carrying out the computations necessary for the verification procedure (including the encryption of outgoing messages and decryption of incoming messages intended for this particular device) and of reliably communicating with the AVP **106** over the Agreement Communication Channel **103**. Each device is also registered at the AVP **106**, together with the key derivation parameters stored in the device (e.g., pseudorandom number generator and its seed, etc). In addition, the association between the AP users and their devices is also stored in the User and Device Database **114** at the AVP **106**.

[**0059**] It is possible to allow the cases where each device may have multiple AP users associated with the device or each AP may be associated with multiple devices. Depending on the requirements the application of the present invention, the multiple users per device may or may not be permitted. For instance, if a particular application of the present invention issues one and only one device for each registered AP user, it is no longer be necessary for the AVP **106** to distinguish the user from the device and the data items for each user may be stored together with the data items for the device issued to the user. During normal operations, the system **100** may use the identifier of either as a reference to locate these data items. This results in more efficient processing than in the multiple user case.

[**0060**] The User and Device Database **114** is also used to log and store the records of each agreement session by recording the SAS messages to and from the agreement parties **101**,**102** and the AVP **106**. Each such agreement transcript can be accessed by the user, device or transaction IDs. This can be used to prevent replay of transactions by reusing a timestamp and to resolve potential claims regarding the verification procedure and the parties involved.

[**0061**] Security Protocol

[**0062**] The security protocol, termed the Secure Agreement Submission Protocol (SAS), is explained in more detail in this section. As part of the description the terms used in the protocol are defined.

[**0063**] Device ID (DID): A unique identifier for each AP (client) Device involved in the agreement generation, transmission, authentication, and verification. This ID is public in the sense that it may be included in messages as plain text, i.e., in non-encrypted form and that it is placed in the non-encrypted part of the message. It can also be used as the address of the device during communication. For instance, the physical address of the network interface (MAC address) of the device can be used for this purpose.

[**0064**] User ID (UID): A unique identifier for each registered AP entity involved in the agreement. That is, the human or entity using an issued AP client device involved in the agreement generation, transmission, authentication, and verification. This UID is used to identify the current user of an AP client device and there is a one-to-one mapping between the UID and an account opened at the AVP **106**. This piece of information is private in the sense that the UID must not be transmitted in plaintext during the protocol execution. Examples of a UID include a name, an e-mail address, a driver's license number, or some account id. The UID is only needed in case the client device has multiple users and is needed to identify the specific user (of many) of the device that is attempting the transaction. The UID may or may not be stored on the device depending on the security needs. If the device has only one registered user, the UID is unnecessary, thus allowing to not store any user-identifying information of the device at all.

[**0065**] Private Identification Entry (PIE): The shared secret input by the user. It is entered by the user whenever the user attempts a transaction. Preferably it is issued to the user following the registration of the user for the application

that the client device is used for. It can also be selected by the user at such time. The PIE is an alphanumeric string. In order to speed up the user entry to make it easier for the user to remember it, the PIE can be a number such as 4-digit or 5-digit PIN. It is a piece of highly secure information in the sense that it is never transmitted during the protocol execution, it is only known to the user and the AVP **106**, and its secrecy should be well protected. It is assumed that the PIE can be input by the user on an AP device in a secure fashion or it may be deterministically generated using a biometric device such as a fingerprint sensor. For example, a computation applied on the fingerprint data received from the fingerprint sensor can be used to generate a PIE that is initially communicated to the AVP by the user. Whenever the user attempts a transaction, the user applies her finger to the fingerprint sensor, thus generating the PIE. The PIE is not kept in permanent storage on the AP device, but is used as an intermediate parameter required for the generation of the encryption key for a transaction and it should not be retained by the device for a period longer than the transaction execution time. If a particular implementation of the present invention uses a form of PIE that is not convenient for a user to input for each agreement transaction and the device needs to store its user's PIN, the storage must be secure and tamper-resistant. The user's PIE is also stored in the User and Device Database at the AVP, which is considered to be a secure facility.

[**0066**] Device User ID (DUID): An identifier for each device to locally identify its users, if the application of the present invention assigns multiple users to a single AP device. The mapping between the DUIDs of a particular device and the assigned users' UIDs is stored in the record of that device the User and Device Database at the AVP, as well as at the device itself. At the same time as a user inputs her PIE at an AP device, she shall also supply her DUID. The DUID is public in the sense that it may be transmitted as plaintext in messages. The DUID of the current user may be stored at the AP device during the execution of a transaction.

[**0067**] Digital Signature (DS): A digital signature associated with a message can be used to verify that a document has not been tampered with and that it was generated by the signer. For a given block of data, a message digest (MD) can be computed using a digest algorithm such as a Hash function. The resulting digest is then encrypted using the encryption key of the signer and the resulting encrypted block of bits is the signature. In order to verify a signature, the recipient decrypts the signature using the key of the sender. If the receiver generates a digest value from the received message which matches with the digest decrypted from the received signature, then the signature is accepted as valid and the received message is considered to be the original unaltered message.

[**0068**] Random Sequence Number (RSN): The RSN is a pseudorandom number that is generated from a locally stored pseudorandom sequence number function R (a pseudorandom number generator). Such RSN functions are well known in the art. Typically the generation of a pseudorandom number also involves another parameter, a seed S. The seed is used as the initial input parameter for the generator R to generate its first pseudorandom number output. From then on, the generator uses the output from the previous iteration as the input for generating the new pseudorandom number. In the SAS of the present invention, the RSN number may be generated either by an AP device or the AVP. Each AP device has its own R and S, which are securely stored on the device and at the AVP. On the AVP, given the DID of an AP device by which a RSN is generated, a program can deterministically locate the same pseudorandom number generator function R and the corresponding pseudorandom number generation seed S for that device from the User and Device Database containing information about all issued devices.

[**0069**] Timestamp (TS): The time associated with a transaction. It can be generated from a reading from a per-device local clock or delivered to the device on a per transaction basis. For example, if the device is used in a purchasing application, the TS can be the TS of the purchase order that the merchant and the consumer will agree on. The TS should be an element of an increasing sequence of values with a known and generally long period between repetitions of values. It is used for two purposes: as an indicator of a device's local time and as a parameter to control the pseudorandom sequence number generator of the same device. In the former case, the TS is used to prevent message replay, as no two messages from a given source should have the same TS. In the later case, the TS is used to control the number of iterations of the generator R before the final output is used as the next pseudorandom number by the SASE.

[**0070**] Transaction: The complete execution of one agreement transmission, authentication, and verification. On an AP Device, a transaction begins when the device generates its view of the agreement and ends when a receipt from the AVP is received and understood. A specific application might include multiple such transactions in order to accomplish the goal of the application. For example, if the application is that of a consumer purchasing goods or services from a merchant, a first transaction might be that of acknowledging and pre-authorizing the purchase and a second transaction might be that of confirming and authorizing the purchase after the completion of the first transaction (when an adequate response is received from the AVP)

[**0071**] Transaction ID (TID): A unique identification number assigned to an agreement. The method of generating the TID is generally application specific and it can be generated by one of the agreement parties or a component of the AVP, such as the View Gathering Module. The Gathering Module will use the TID and an additional parameter, Number in Transaction (NIT), that specifies the number of parties in the agreement, to identify when it has received a complete set of views for an agreement. In a two-party agreement, the TID and NIT may not be required.

[**0072**] View: The processed agreement data by an AP device. A view of an agreement consists of an encrypted portion and an unencrypted portion. The encrypted portion includes reference information (the other party's Device ID, and optionally the User ID, a message digest MD, which can also be digitally signed) and the specific agreement data. The unencrypted or plaintext portion consists of reference information including Transaction ID, Number in Transaction, Time Stamp, Device ID and Device User ID.

[**0073**] Agreement Data: The agreement data conveys the specific details that are agreed upon by the involved parties. For example, the amount that one party agrees to pay a second party is a agreement data. Agreement data may also contain information that is relevant to the agreement, but

needs to be shielded from the other agreement parties. For example, the financial account with which one party agrees to pay the second party may be included in the agreement data, but this is not protected from the second party through encryption. The agreement verification module will be configured to determine that both parties agree on the amount and the participants, while protecting and delivering the other agreement data, such as the account information for the appropriate additional processing, such as by a Transaction Processing Component **116**. The primary purpose of the SAS and the cryptographic algorithm is to protect the agreement data during transmission and to shield the other information from the other agreement parties, while providing the security properties of privacy, authentication, user anonymity, non-replayability and non-repudiation

[**0074**] The method **200** of encrypting an SAS view of the present invention, referred to as the SASE, is illustrated in **FIG. 2**. The SAS view **210** illustrated in **FIG. 2** corresponds to an AP View **110**, **120** of **FIG. 1**. As shown in **FIG. 2**, an AP view **210** includes a cipher text part (or encrypted part) **212** and a plaintext part **214**. The plaintext part **214** includes the TID, the NIT, the DID of the AP device generating the view, the local TS of that AP device, the DUID of the current user of the device, the TID and the number of parties in the agreement. The encrypted part **212** includes four fields: the digital signature DS **216**, the agreement, the UID of the AP, and the DID of the other AP involved in the agreement. The DID of the other AP involved in the agreement is the minimum necessary reference field in order to provide the desired properties of the SAS protocol. The DS further increases the strength of the security by ensuring that no other party has tampered with or modified the contents of the view in any way. The TID and NIT are not necessary in a two-party agreement. The purpose of the TID and NIT is to associate views (messages) and responses to these messages and, alternatively, information that relates messages and responses to these messages can be provided as part of the agreement data itself in a way that depends on the particular application.

[**0075**] In the case that the AVP only allows one user to be associated with each device, the UID field may be omitted because the AVP can derive such a UID based on the DID. The UID of the other party involved in the agreement is not included in any view, so that the other AP involved in the agreement may remain anonymous. The DUID field is also not necessary in this case.

[**0076**] At first, the DID **234** of the view generating device and the TS **236** obtained from the device's local clock or provided as a part of the agreement data, are input to the device's pseudorandom number generator **252** to generate a RSN **246**. In the SASE, the TS **236** is used to control the number of iterations of the pseudorandom number generator **252**. Only the final result after these iterations is used as the output RSN **246** for the SASE.

[**0077**] There are several variations in how the TS is employed to generate the RSN. One method of using the TS to control the number of inductions is to use the difference between the TS value (in number of minutes or seconds) and another mutually agreed base time value as the number of inductions. The generation of RSN is denoted as: RSN=R (S, TS, T_0) where T_0 is the base time. The base value T_0 is stored both at the AP and the AVP which will store the base value in the User and Device Database in the record for the AP device and is specific to each AP device. The mutually agreed base time is advanced on both the AP device and the AVP in order to reduce the number of inductions to produce a SASE RSN, as long as the advancement of the base time on AP and AVP can be synchronized. If desired, as the base time advances, the seed may also be updated. For example, the new seed S' may be the S'=R (S, T_0', T_0) where S is the original seed, T_0 is the original base time, and T_0' is the new base time. The property of the SASE that needs to be maintained is that given a particular sender's pseudorandom sequence number generator R, its seed S, and the same TS value as used by the sender, the receiver can deterministically reproduce the same RSN as was generated by the sender

[**0078**] A hash function H **254** is then applied to the output of two-argument function F that when applied to the locally generated RSN **246** and the PIE **248** input by the AP user outputs a single argument (typically a string), in order to create the encryption key K **250**:

[**0079**] K=H (F ((PIE, RSN)) or further expanded to: K=H (F (PIE, R(S, TS, T_0))).

[**0080**] Such Hash functions are difficult to invert and are well known in the art. The function can by any known function, such as a function that appends the PIE string to the RSN string, or XOR's the PIE and the RSN, etc.

[**0081**] A message digest function **258** is applied to the data, the UID of the AP user, and the DID of the other AP involved in the agreement to generate a message digest (MD) **216** of the view. The message digest function **258** can be a hash function that takes as input the plaintext of these three data items and produces a single number. Such hash functions for use in producing message digests are also well known in the prior art. For example, the hash function SH1 is often used for this purpose.

[**0082**] The encryption algorithm with the encryption key K **250** is then applied to the message digest **216**, the agreement data **244**, the UID of the AP user **240**, and the DID of the other AP involved in the agreement **242** to generate the cipher text part **212** of the view. The DID **234** and TS **236** which were used to generate the encryption key are also included in the view as plaintext. The TID **230** and NIT **232** are also included in the plaintext part **214** of the view. Thus, the agreement view **110** from the first AP device is the following:

[**0083**] AP View 1={TID, NIT, DID_1, TS_1, $DUID_1$, Encryption [K_1: (UID_1, DID_2, data, MD_1)]}

[**0084**] The specific encryption algorithm employed by the system **100** can be any of the known symmetric key-based encryption algorithms chosen to provide sufficient protection. However, the present invention includes the key generation process to be used with the chosen encryption algorithm.

[**0085**] As one embodiment of the SASE, the encryption algorithm **256** is TripleDES, the Random Number Generator **252** is a Mersenne Twister, the seed is a 32-bit number, the time-stamp is a 64-bit number representing seconds, the PIE is four digits, and the Hash function **254** is SHA1 and the function F that generates the input to the Hash function, is a function that appends the PIN to the RSN.

[0086] For further protection, the SAS protocol uses message padding in order to further prevent "known-text" attacks. In "known-text" attacks, an attacker who knows the plaintext of the agreement will attempt to reverse engineer the encryption key and eventually, with enough successful attacks, the other parameters used by the key derivation process. If successful, the attacker becomes capable of reproducing the encryption key for that particular view. Since the key changes over time (each timestamp is associated with a new key), this attack would reproduce the key for that particular timestamp only. Further transactions using the same timestamp are denied through comparison with the previous transaction timestamps stored at the AVP.

[0087] The padding scheme will insert random bits before and after the real fields so that an observer cannot determine where the real data begins, increasing the difficulty of "known text" attacks. The amount of padding is determined by the lengths of the overall message and the included data. In one embodiment of padding **300**, as illustrated in **FIG. 3, a** padded field **302** starts with a field of fixed length **312**, which describes the number of random bits inserted before the actual encrypted fields. This field **312** is followed by a string of random bits **314** of the length specified by this field **312**, and then the real data field **310**. Random tailing bits **316** are also appended after the end of all encrypted fields to further increase the difficulty for an attacker to extract the real cipher text part of a view. Since the total length of each field is known, it is not necessary to specify the length and offset of the tailing random bits **316**. If the length of each field is not known, field **312** will be followed by an additional field that specifies the offset of the tailing random bits **316**. In another embodiment, random bits are inserted only before and after all fields. In this case although the difficulty for an attacker to determine the location of each data field is reduced the processing of each SASE message is also reduced. Padding is applied before encryption is applied during view construction.

[0088] This completes the description of the SASE mechanism for generation of a secure message by an AP. A similar procedure is defined in a later section for decryption of the message at the AVP.

[0089] View Gathering

[0090] At the AVP **106**, the Views **110,120** belonging to the same agreement transaction but generated by different AP devices will first be gathered together by the View Gathering Module **108** before any further authentication and verification processing. When all the views of an agreement are collected, they are given to the Agreement Authentication Module **118**.

[0091] The SAS of the present invention permits agreement parties to be involved in multiple, simultaneous transactions with differing parties. In addition, multiple transactions from differing parties can also be simultaneously active at the AVP **106**. In general, the view gathering function decides which views belong to the same agreement transaction and at what point the gathering is completed so that all views belonging to the same agreement transaction can be forwarded to the authentication module **118**. A TID must be used to tag each view of an agreement so that the gatherer can match the views belonging to the same agreement and process them together.

[0092] The View Gathering Module **108** uses the TID in each message to match the views. When the View Gathering Module **108** has collected the proper number of distinct views, given by NIT, the View Gathering Module **108** will forward the set of views to the Authentication Module **118**. The parameters TID and NIT are sent in plain text so that the View Gathering Module **108** can operate on the views prior to authentication and decryption. This permits greater flexibility in that the View Gathering Module **108** can be physically separated from the AVP **106**. In order to insure the integrity of the TID and NIT, the TID and NIT are repeated in the agreement data. For this purpose, the TID and a list of DIDs of the AP devices involved in the agreement are included in the encrypted portion.

[0093] In alternative implementations, the TID and NIT are only included in the encrypted portion of the message and must be decrypted and authenticated (by the Agreement Authentication Module) prior to handling by the View Gathering Module. In this case, the View Gathering Module holds the decrypted views until a complete set is obtained.

[0094] The View Gathering Module **108** holds unmatched views of a Transaction for a maximum period of time, called the Transaction Time-out period. After this time has elapsed without collecting a complete set of views, the views are discarded and, optionally, the agreement parties are notified.

[0095] Decryption

[0096] The views **110,120** are decrypted at the AVP **106** by the Agreement Authentication Module (AAM) **118**.

[0097] **FIG. 4** shows a detailed explanation of the procedure followed by the AAM **118** and the Agreement Verification Module (AVM) **112**. More particularly, **FIG. 4** shows a method **400** of decryption of the above-mentioned AP View 1 **110** and AP View 2 **120**, into decrypted AP View1**410**, which includes in plaintext TID, NIT, TS1, DID1, DUID1, and decrypted AP View2 **460**, respectively which includes in plaintext TID, NIT, TS2, DID2, DUID2.

[0098] Initially, when the views **110** or **120** are received, it is useful for the AAM **118** to check the validity of the TS of the views. This operation may prevent attacks conducted by changing an AP device clock or replaying an intercepted view. For this purpose, the AVP **106** stores a clock offset value for each AP device **101,102** in its User and Device Database **114**. This offset describes the difference between the device **101,102**'s local clock and the system clock of the AVP **106**. With the offset and the TS, the AVP **106** can verify if the message generated by such a device **102, 104** occurs within a reasonable time-window before the message arrives at the AVP **106**. Only messages generated during this period are accepted. Otherwise an "Expired Transaction" error message is generated and sent back to the APs using a method described later in this section. The size of this time window, and the accuracy of the clocks would depend on the requirements set by the application of the present invention.

[0099] Referring now to **FIG. 4**, when the AAM **118** is decrypting a transaction view message **110** from a client **101**, based on the plaintext DID field **430** of the view **110** the AAM locates the corresponding pseudorandom sequence number generator R **434** and seed S for the device **101** which generated the received view **110** using the User and Device Database **114**. Then using the TS **432** also contained in the AP View **110** as plaintext, the AAM can inductively reproduce the RSN **436** which is identical to the RSN **246** (of **FIG. 2**) used during the derivation of the encryption key.

Because the TS value which is required for the AAM to determine the RSN of the view generating AP device **101**, **102** is enclosed in each message, it is not necessary for the AAM **118** and the AP devices **101**, **102** to have synchronized clocks for RSN derivation purposes.

[**0100**] The AAM **118** then locates the current user of the AP device **101** in its User and Device Database **114** using the DUID field **433** of the view. By looking into the record for the AP's current user, the AAM **118** finds the corresponding PIE **438** of the user. Then the AAM **118** reconstructs the encryption key **442** (**250** of **FIG. 2**) used for generating the view **110** by using the same Hash function **440** (**245** of **FIG. 2**) used by the AP. With the encryption key known, the AAM can decrypt the full view message contained in the view **110**. After the decryption, if random bit padding was applied during the construction of the view, the padding bits are removed to reveal the true data fields. After the encrypted fields are decrypted, the UID **422**, the DID of the other party **424**, and Data **426** are fed into a digest algorithm **446**, which is identical to the digest algorithm **258** used by AP device, to produce a digest **448**. This digest **448** is then compared with the MD **428** resulted from decrypting the digital signature contained in the received view. Only if both digests are the same, the digital signature is considered correct. Otherwise, the view is considered altered from the original. The same procedure takes place for the received AP view **120** in order to ensure that MD2 **478** corresponds to data **476**.

[**0101**] If the AAM **118** is not able to successfully decrypt the message or the digital signature is not correct, then the authentication is deemed to have failed. The AP's will be notified through an "Authentication Failed" response message.

[**0102**] The above described SASE encryption scheme and key generation method is also used by the AVP **106** to encrypt response messages such as errors or, acknowledgements or receipts that are sent back to APs **101**, **102**. In general, the response can also contain arbitrary application specific data. For example, it can be used to transmit special tokens generated by the Transaction Processing Component **116** for AP users for later use.

[**0103**] Specifically, using the same basic SASE encryption method, to send a response message to AP_i, the AVP will use the destination AP parameter DID_i to determine the random number generator R_i, the $Seed_i$ and a TS determined by the AVP to generate the RSN. Next, the destination APs current user's PIE_i RSN and Hash function are used to generate the encryption key K. A Response Message to AP_i has the following fields and is formatted as:

[**0104**] $ResponseMessage_i$={TID, DID_i, TS, $DUID_i$, Encryption [K: (MD, data)]}.

[**0105**] $ResponseMessage_1$, is then transmitted to APi. When received, APi is able to use the plaintext parameters in the message and its internal parameters to derive the decryption key and decrypt the message. During this process, the AP device may use the included DUID to prompt its user for a PIE if the PIE is not cached at the device.

[**0106**] In certain situations, because we are using a symmetric cryptography algorithm, in which the same key K derivation procedure can be carried out by either side, the above described AVP response message can be generalized for carrying arbitrary application data in messages.

[**0107**] When used for sending error messages and receipts back to the APs, the return messages are sent in a reversed path along the Agreement Channels to the APs. If the views are sent separately from each APs (via gathering function) to the AVP, the return messages are also sent independently to the destination APs. Such reverse communication does not need to go through the view gathering module. However, each return message does need to include sufficient information, such as the agreement TID in the message, so that the receiving AP device can identify to which agreement transaction the return message belongs.

[**0108**] Agreement Verification

[**0109**] After both views **110**, **120** are successfully decrypted, the AVP **106** verifies the agreement using the Agreement Verification Module **112** that executes a procedure consisting of a list of matching rules to be applied to the agreement views. A series of basic matching operations between the fields in the views **110**, **120** are carried out and then optionally, application specific matching rules can be applied. The basic matching operations are illustrated in **FIG. 4** and include:

[**0110**] The DID included in each view's plain text part matches with the DID of the other party included in the other view's encrypted part. That is, 416 matches with 474 and 466 matches with 424.

[**0111**] The UID included in each view's cipher text party matches with the current user of the view generating device as determined by the view generating device's device ID and the current user's DUID. That is, the user ID derived from DID_1 **416** and $DUID_1$ **420** should matches with UID_1 **422** included in the encrypted part of the view. The same matching rule applies to DID_2 **466**, $DUID_2$ **470** and UID_2 **472**.

[**0112**] The Transaction ID, TID **412** (or **462**), of each view is matched with the TID **462** (or **412**) of the other party. In addition, the plaintext NITs are verified by counting the listed DIDs in each view.

[**0113**] If one of the matching rules is fails during the examination, the verification process is stopped and "Verification Failed" error messages are sent back to both APs using the return message method described earlier. For example, error messages are generated as the following with error1 and error2 being an error code or a descriptive message which both the APs and AVP can understand:

[**0114**] ErrorMessage1={TID, DID_1, TS_1, $DUID_1$, Encryption [K:(MD, error1)]}

[**0115**] ErrorMessage2={TID, DID_2, TS_2, $DUID_2$, Encryption [K:(MD, error2)]}

[**0116**] The next step is for the AVP to verify that the agreement data included in each view's cipher text part matches with each other according to the needs of the application. The SAS is a submission vessel protocol for agreements. Thus it does not define the format and specification of the agreements it carries. Therefore, to accommodate the application in determining whether two agreements really semantically agree with each other, an interface is provided by the AVP so that each application may provide its

own additional agreement verification rules for verifying that the agreements included in the views are consistent with each other.

[0117] For example, a simple application independent plug-in procedure that can be used is a bit-matching function. If two agreements are exactly the same, bit by bit, the matching test is passed. More complex plug-ins may involve application specific cryptographic operations and semantic correspondence.

[0118] The Agreement Verification Module **112** may be physically implemented on the AVP, together with the authentication processing implementations. Alternatively, the verification process can be implemented on a different device but able to communicate with the other modules in the AVP through a reliable and secure communication channel.

[0119] At the completion of the verification process, the AVP may forward the agreement data decrypted from received views to a Transaction Processing Component **116**. However, in this case the communication between the AVP **106** and the Transaction Processing Component carrying out the verification processing must be secure, if not co-located. From the SAS perspective, the agreement data extracted from each received view is already verified by the AVP.

[0120] Because of the additional communication, a timeout mechanism may be included so that if no reply is received from the Transaction Processing Component **116** process within a certain time, the AVP **106** sends error messages back to the APs **101102**

[0121] When in an application of the present invention the Transaction Processing Component **116** is physically located on a different device than the AVP **106**, the application may employ additional cryptography techniques to offer additional privacy features. For example, each AP may apply additional encryption to the agreement data before it applies SAS encryption. This pre-encryption can only be decrypted by the Transaction Processing Component **116** process, which is not co-located with the AVP. Thus, even the AVP will not be able to discover the contents of the agreement beyond the information needed for basic matching.

[0122] At the end of the verification process, application specific receipts may be generated for the AP's **101**, **102** describing the result of the verification.

[0123] $ReceiptMessage_1$={TID, DID_1, TS_1, $DUID_1$, Encryption [K_1:(MD, $receipt_1$)]}

[0124] $ReceiptMessage_2$={TID, DID_2, TS_2, $DUID_2$, Encryption [K_2: (MD, $receipt_2$)]}

[0125] The receipts are sent back to the APs using the method for the AVP to send messages back to APs, as describe earlier. It is important to point out that the contents of the receipts do not need to be understandable by the components of the AVP. This is different from the error messages generated by the authentication process of the AAM. The reason for this distinction is to separate the results from authentication processing from the results from the agreement verification processing. This separation gives the applications of the present invention more capability to include additional features. For example, when there is an additional Transaction Processing Component **116** that is physically separated from the AVP, the agreement verification process may include confidential information in its receipts. It is not necessary to allow the AVP to understand the contents of the receipts.

[0126] The departure from the AVP of the receipt or error message for the last AP involved in the agreement marks the end of an agreement authentication and verification transaction at the AVP **106**. The arrival of a receipt at an AP **101102** marks the end of an agreement authentication and verification transaction at the AP.

[0127] AP View 1 **110**, AP View 2 **120**, and Agreement Verification **106** are implemented in respective software programs which, when executed by a computer, cause the computer to execute the respective functions described herein above. Each of the programs can be stored on a computer-readable medium.

[0128] Extensions of the SAS Protocol

[0129] The above SAS protocol description is presented for agreements between two parties. However, the SAS protocol of the present invention can be extended for agreements involving more than two parties. In this case, for a transaction involving n parties, the transaction view message from the i-th participant is:

[0130] $ViewMsg_i$={TID, NIT, DID_i, TS_i, $DUID_i$, Enc [K_i:(MD_i, TID, UID_i, DID_0, . . . , DID_{i-1}, DID_{i+1}, . . . , DID_{n-1}, agreement)]}

[0131] Correspondingly, the verification and authentication rules are:

[0132] $ViewMsg_0.DID_i$==. . . ==$ViewMsg_j.DID_i$**32**=. . . ==$ViewMsg_{n-1}.DID_i$, where i=0 . . . n-1

[0133] For all i's (i [0, n-1]), using $ViewMsg_i.UID_i$ and $DUID_i$ to search the User and Device Database for the reference UID. This UID should be the same as UID_i included in the encrypted part of the $ViewMsg_i$.

[0134] $ViewMsg_0$.TID== . . . ==$ViewMsg_j$.TID== . . . $ViewMsg_{n-1}$.TID, where i=0 . . . n-1

[0135] $ViewMsg_0$.NIT== . . . ==$ViewMsg_j$.NIT== . . . ==$ViewMsg_{n-1}$.NIT, where i=0 . . . n-1, and NIT is equal to n, the number of parties listed in the agreement.

[0136] The submission methods of the views in a two AP system are extended to agreement transactions involving more than two APs. If the view gathering and generation processes are separated, exactly the same methods used by a two AP system can be used for a system with more than 2 APs. The View Gathering module collect views from all parties in the agreement using the TID and NIT included in the message.

[0137] When the view gathering function is implemented separately from the view generation function, the view gathering function can be physically implemented at an external device (in which case the APs send their views to this view gathering device then the view gathering device forwards all views together to the AVP.

[0138] Alternative View Gathering Methods

[0139] In an alternate version of the invention, called integrated view gathering, the view gathering mechanism is distributed to the APs so that the views are collected

sequentially by successive agreement parties as they are transferred to the AVP. If the view gathering and generation are integrated in this manner, a submission chain needs to be set up beforehand among all APs. After the first AP on this chain generates its view, the view is sent to the second AP in this chain. Upon receiving a view from the first AP, the second AP is triggered to generate its own view. Then both views are forwarded to the third AP in this chain, and so on. This process is executed in turn by each AP on this submission chain and finally all views are sent by the last AP on the submission chain to the AVP. In that case, the TID and NIT can be omitted also.

[0140] An example of such an integrated view gathering and generation system is shown in the computer system **500** of **FIG. 5**. As shown in **FIG. 5**, the first AP device **502** comprising a local Agreement Channel **505** generates its view **522** of the agreement. The view **522** is sent to the second AP device **504** via the local agreement channel **505**. Upon receiving the view **522** from the first AP device **502**, the second AP device generates its view **524** of the agreement. Then both views **522524** of the agreement are sent to the AVP **506** via an agreement channel **503**. In some implementations, the views may even be concatenated together and sent as one message. In this variation, because the views are gathered as they are generated, it is no longer necessary for the system **500** to include a View Gathering component. The AVP **506** itself comprises three components the Agreement Authentication Module **510**, which is identical to the Agreement Authentication Module **118**, the Agreement Verification Module **512** which is identical to the Agreement Verification Module **112**, and the User and Device Database **514**, which is identical to the User and Device Database **114**.

[0141] Another variation of the invention permits the assembly of a multi-layered agreement view as a result of an integrated view gathering architecture. In this system, each successive AP may perform an operation on the agreement data received from APs earlier in the chain. The initial agreement data is included in the view of the first AP. The second AP uses the view received from the first AP as part of its own agreement data and produces its own view, based on a function of the received view. Finally, what the AVP receives is a single, multi-layered view. Combined with the physical separation of AVP modules, such as the MM and AVM and appropriate encryption/decryption algorithms, applications of this variation of the present invention can support new capabilities in supporting privacy. An example application of the present invention variation for electronic voting will be given later in this document.

Examples of Applications of the Present Invention

[0142] The first application example of the present invention is shown in **FIG. 6**. It is a wireless payment system **600** for payments by consumers in physical retail stores. The architecture is similar to that shown in the chained integrated view gathering variation shown in **FIG. 5**. In this example, the backend server called Secure Transaction Sever (STS) **610** is the AVP. The STS **610** is further connected to a Transaction Processing Component that is a Financial Institution **612** to carry out the actual processing of the financial transactions. The APs are the consumers and the merchants and they have their own AP devices **602** and **604**. For consumers, the AP device **602** can be any mobile device with wireless capability, such as Personal Digital Assistant, a mobile phone or a credit card sized mini-computing device which are capable of wireless communication and carrying out SAS computations. For merchants, the AP device can be a computer **604** comprising a wireless LAN access points **606** providing service to a WLAN service area **614** and a connection to the backend STS **610** via the Internet (called an Agreement Channel **608**).

[0143] The agreement is the data requesting a monetary transaction between the consumer and the merchant for purchase of physical or virtual goods. After the consumer finalizes her purchase, her AP device **602** generates her view of the transaction. The view is sent to the merchant device **604** using a wireless LAN access service, which in turn triggers the merchant device **604** to generate the merchant's view. Then the merchant device sends both views together to the STS **610** over the Agreement Channel implemented as a secure Internet connection. After the STS **610** authenticates the identities of both the merchant and the consumer through decryption, it extracts the monetary transaction request data from the views and performs the basic verification procedures. If successful, the STS forwards the requests to a financial institute **612** for further transaction processing and eventual monetary exchange. Results from the financial institute **612** are returned to the STS **610** and encrypted as receipts to both the merchant and consumer. Both receipts are sent to the merchant device **604** over the Agreement Channel and the merchant device **604** forwards the consumer receipt to the consumer device **602** over the wireless LAN. In a variation, the purchase occurs in two stages, the first stage being a transaction during which the merchant and the consumer request a purchase and the second stage being a transaction during which the consumer and the merchant authorize the purchase, with the consumer also selecting which financial account to use for the transaction.

[0144] In this example, the wireless payment application uses an integrated view gathering approach due to the fact that the consumer AP device **602** does not have a direct communication link to the AVP **610** and the merchant device **604** concatenates its view after it receives a view from client device **602**. At the AVP end, the authentication processing and verification processing are co-located on the STS **610**. In addition to the components in the present invention, the application also has the additional Transaction Processing Component of a financial institution **612** to carry out additional application specific processing.

[0145] The second application example is shown in **FIG. 7**. It is an electronic voting system **700**. The system is an application of the variation of the present invention with an integrated view gathering function and a multilayered agreement. The scheme allows wireless voting at official voting stations and provides the following properties: registered voter authentication, voting station authentication, voter anonymity, vote confidentiality, vote auditing, and ensuring one vote per voter. In this system, voters and voting stations are the APs. Their devices **702** and **704** are the AP devices. The Authentication Server **706** and the Voting Center **708** together implement the functions of the AVP and the Transaction Processing.

[0146] The agreement data, consisting of the ballot, is obtained from a voting station by a request from the voter. The voter receives the proper ballot on her device and then

enters the vote information. The filled-in ballot is first encrypted by the voter encryption **710** function of the voter device **702** using the standard encryption technique of the SASE. The encrypted ballot data is then sent to the voting station **704**. The voting station **704** uses its station encryption function **712** to double encrypt the ballot, using the key generated by the voting station. In this voting application, an additional requirement is that the actual encryption algorithm used to encrypt the ballot data needs to be "commutative". That is, it does not matter what order the decryptions are applied to a piece of multi-encrypted data, as long as the corresponding decryptions of all encryptions used to produce the multi-encrypted data are applied, the original plaintext can be revealed. Many modern symmetric stream ciphers do fall into this category and thus can be used by this system.

[**0147**] After the double encrypted ballot is sent to the Voter Authentication Server **706**, with the help of information stored in the Voter Database **716**, the Voter Authentication Server **706** uses the Voter Authentication function **714** to remove the encryption applied by the voter device **702**. The resulting ballot, still encrypted with the voting station's key and therefore not readable by the Voter Authentication Server **706**, is then forwarded to the Voting Center **708**. The Voting Center **708**, passes the encrypted ballot to the Station Authentication Server **718**. The Station Authentication Server **718**, with the help of the Voting Station Database **722**, removes the encryption applied by the voting station to reveal the contents of the ballot. The ballot and some additional Information contained in the views are passed to the Ballot Verification server **720**. This applies some specific matching rules to verify that the ballot is consistent. If the ballot is verified, then the plaintext ballot is passed to the Vote Processing function **724** for vote counting and other voting information collection.

[**0148**] By applying the variation of the present invention and commutative encryption algorithms, the voting system is able to authenticate voters (by the Voter Authentication Server **606**) while still maintaining the anonymity of the ballots while collecting vote information (by the Voting Center **608**) from legitimate voting stations. During the process, no single component (other than the voter device) is able to discover what a particular voter voted for or to which voter a fully decrypted vote belongs to.

[**0149**] Tokens

[**0150**] Another application of the SAS is to provide a method of securely distributing special messages called "tokens" that can be thought of as tickets. Such tokens are generated by the AVP as the result of an agreement and sent to one or more members of the agreement. They can be used by members of a previously authenticated agreement to authenticate the other members of the agreement directly without contacting the AVP at the time of authentication. A second use is to authenticate the presentation of the result of a previously authenticated agreement by a third party (who may or may not be a party to the original agreement) without directly contacting the AVP at the time of authentication. The tokens can be used as tickets where in the former case, the identity of the ticket holder and the ticket are important (as in airline tickets), and in the later case, the identity of the ticket holder is not important, just the validity of the ticket. The token should only be used once, as there is not strong security between the two parties.

[**0151**] Let AP1 and AP2 be two parties of an agreement that has been verified by the AVP. At some time in the future, AP2 would like the ability to verify the identity of AP1 without consulting the AVP again. The token is a type of AVP response message in which the agreement data portion of the response message contains special token identifying information.

[**0152**] **FIG. 8** illustrates a method **800** of using the present invention to generate 3rd-party verifiable tokens.

[**0153**] As shown in **FIG. 8**, tokens are generated by the AVP in pairs, with one called token **801** and the other called token receipt **821**. The token **801** is sent to AP1, the party to be authenticated, while the token receipt **821** is sent to AP2, the party that wants the authentication service.

[**0154**] The formats of the token and token receipt are shown in **FIG. 8**. Both are formatted in the same fashion as other AVP response messages. The plaintext part of both token and token receipt contains the same fields as other AVP response messages as described before. Specifically, the plaintext part of token **801** includes DID1 **802**, TS1 **804**, DUID1 **806** and the plaintext part of token receipt **821** includes DID2 **822**, TS2 **824** and DUID2 **826**. The cipher text part of a token **801** contains a token identifier TKID **808** that is used to uniquely identify a token pair, the DID **810** of AP2, a token code **812**, and other data **814** associated with the token. The cipher text part of a token **801** is encrypted by the AVP using a key generated using standard SASE for the current user of AP1. A token receipt **821** is formatted almost the same as a token except for two differences. The first difference is that the token code **832** included in the token receipt **821** is firstly encrypted using SASE with AP1's parameters except for the timestamp. The timestamp could be any future time value TSv chosen by the AVP. Such a TSv **829** is also included in the cipher text part of the token receipt **821**, which is the second difference between a token and a token receipt.

[**0155**] Upon receiving a token, AP1, the party whose identity is to be verified, will decrypt the token and store the TKID **808**, DID2 **810**, Token Code **812**, and token data **814** for future use. AP2, the verifying party, stores the TKID **828**, TSv **829**, DID1 **830**, token code **832**, and token data **834**. The token code **832** stored by AP2 is still encrypted by SASE using the parameters for AP1 and TSv. On the other hand, the token code **812** stored by AP1 is in plaintext form.

[**0156**] At the time of token verification, AP2 requests that AP1 deliver the token to AP2 by sending a Token Request message containing the TKID **828** and the TSv **829** of the token. AP1 receives the request, encrypts the token code **812** with its own SASE parameters and TSv as timestamp value. Then AP1 transmits the encrypted token code to AP2. At AP2, if the received encrypted token code is found to be the same bit by bit as the locally stored token code **832**, the token is verified and thus the user is authenticated as being a member of the agreement.

[**0157**] For the second case, where the identity of the token holder is not important, the original token holder can pass the encrypted token to a third party. Let AP1 be the original token holder and AP2 be the verifier. The third party, P, must store the encrypted token and the necessary parameters, such as TKID, TSv, DID2. P presents the token to AP2, by sending an unencrypted message to AP2 containing the TKID, TSv and the token (encrypted by AP1).

[0158] The tokens are useful to verify that a party has a valid result of an agreement. For example, a party has used a mobile computing device to wirelessly purchase movie tickets and has wireless transmitted one ticket to a companion. When the tickets were purchased, a user receives on her device an encrypted token for each ticket and some additional data such as total number of tickets, time, place, etc. The movie theatre also receives the token information. At entry time, each user wirelessly presents one or more tokens and is granted entry.

[0159] The system also includes permanent or removable storage, such as magnetic and optical discs, RAM, ROM, etc. on which the process and data structures of the present invention can be stored and distributed. The processes can also be distributed via, for example, downloading over a network such as the Internet.

[0160] The many features and advantages of the invention are apparent from the detailed specification and, thus, it is intended by the appended claims to cover all such features and advantages of the invention that fall within the true spirit and scope of the invention. Further, since numerous modifications and changes will readily occur to those skilled in the art, it is not desired to limit the invention to the exact construction and operation illustrated and described, and accordingly all suitable modifications and equivalents may be resorted to, falling within the scope of the invention.

What is claimed is:

1. A computer system securely transmitting and verifying a two-party agreement, said system comprising:

a first device, operated by the first party, developing and transmitting a first view of the two-party agreement, the first view including an encrypted part and an unencrypted plaintext part;

a second device, operated by the second party, developing and transmitting a second view of the two-party agreement, the second view including an encrypted part and an unencrypted plaintext part; and

a third device comprising a verification party receiving the first view and the second view, decrypting the encrypted part of the first view and the encrypted part of the second view, affirming that the first view was developed by the first party and that the second view was developed by the second party, comparing the decrypted part of the first view with the decrypted part of the second view, and transmitting a signal based on the outcome of the comparison.

2. The computer system as in claim 1, wherein either the first device transmits the first view to the third device and the second device independently transmits the second view to the third device, or the first device transmits the first view to the second device and the second device concatenates the first view and the second view and transmits the concatenated view to the third device.

3. The computer system as in claim 2, wherein the third device transmits a response to the first device and the second device either if the third device determines that the decrypted part of the first view matches the decrypted part of the second view for the purposes of the agreement or if the third device determines that the decrypted part of the first view does not match the decrypted part of the second view for the purposes of the agreement.

4. The computer system as in claim 3, wherein the plaintext part of the view of the first device includes the device identification of the first device and the transaction timestamp of the first device, and the encrypted part includes data corresponding to an agreement and a device identification of the second device.

5. The computer system as in claim 4, wherein the plaintext portion may optionally include a transaction ID and the encrypted portion may include one or more of a transaction ID, a message digest, a transaction timestamp, the user ID of the user of the first device and padding data.

6. The computer system of claim 5, in which the encrypted part of the view from the first device is encrypted using a symmetric secret encryption key, and a cryptographic algorithm in which the key and cryptographic algorithm are known only to the first device and the third device and the keys are not communicated either between the first device and the second device or the first device and the third device.

7. The computer system of claim 6 in which the encryption key used by the first device to encrypt the view is generated based on a secret value comprising a Private Identification Entry (PIE), and a deterministic algorithm based on a set of stored parameters and functions, wherein the PIE is known only to the operator of the first device and is not stored on the first device and is also known to the third device that receives the encrypted view developed by the first device, and the deterministic algorithm and the set of stored parameters and functions are known only to the first device and the third device.

8. The computer system of claim 7 in which each of the first device and the third device comprises a known pseudo-random number generator, a known hash function and a known transformation function in order to deterministically produce identical encryption keys given the same transaction timestamp for purposes of encrypting the first view by the first device and decrypting the first view by the third device, wherein:

the pseudo-random number generator receives as input the seed, the base timestamp and the transaction timestamp and produces as output a random number;

the transformation function takes as input the said random number and the PIE and produces as output a number; and

the hash function takes as input the said number and produces as output the encryption key.

9. The computer system as in claim 3, wherein the response from the third device to the first device comprises a plaintext part and an encrypted part, wherein the plaintext part of the response includes the device identification of the first device and a transaction timestamp and may optionally include a transaction ID, and the encrypted part includes data corresponding to a response to the agreement and may include one or more of a device identification of the first device, device identification of the second device, a transaction ID, a message digest, a transaction timestamp, the user ID of the user of the first device and padding data.

10. The computer system of claim 9, in which the encrypted part of the response of the third device to the first device is encrypted using a symmetric secret encryption key and a cryptographic algorithm in which the key and cryptographic algorithm are known only to the first device and

the third device and the keys are not communicated either between the first device and the second device or the first device and the third device.

11. The computer system of claim 10 in which the encryption key used by the third device to encrypt the encrypted portion of the response to the first device is generated based on a secret value comprising a Private Identification Entry (PIE), and a deterministic algorithm based on a set of stored parameters and functions, wherein the PIE is known only to the operator of the first device and is not stored on the first device and is also known to the third device and the deterministic algorithm and the set of stored parameters and functions are known only to the first device and the third device.

12. The computer system of claim 11 in which each of the third device and the first device comprises a known pseudo-random number generator, a known hash function and a known transformation function and deterministically produce identical keys given the same transaction timestamp for purposes of encrypting the response by the third device and decrypting the response by the first device, wherein:

the pseudo-random number generator receives as input the seed, the base timestamp and the transaction timestamp and produces as output a random number;

the transformation function takes as input the said random number and the PIE and produces as output a number; and

the hash function takes as input the said number and produces as output the encryption key.

13. The computer system as in claim 3, wherein the plaintext part of the view of the second device includes the device identification of the second device and the transaction timestamp of the second device, and the encrypted part includes data corresponding to an agreement, and a device identification of the first device.

14. The computer system as in claim 13, wherein the plaintext portion may optionally include a transaction ID and the encrypted portion may include one or more of a transaction ID, a message digest, a transaction timestamp, the user ID of the user of the second device and padding data.

15. The computer system of claim 14, in which the encrypted part of the view from the second device is encrypted using a symmetric secret encryption key, and a cryptographic algorithm in which the key and cryptographic algorithm are known only to the second device and the third device and the keys are not communicated either between the first device and the second device or the second device and the third device.

16. The computer system of claim 15 in which the encryption key used by the second device to encrypt the view is generated based on a secret value comprising a Private Identification Entry (PIE), and a deterministic algorithm based on a set of stored parameters and functions, wherein the PIE is known only to the operator of the second device and is not stored on the second device and is also known to the third device that receives the encrypted view developed by the second device, and the deterministic algorithm and the set of stored parameters and functions are known only to the second device and the third device.

17. The computer system of claim 16 in which each of the second device and the third device comprises a known pseudo-random number generator, a known hash function and a known transformation function and deterministically produce identical encryption keys given the same transaction timestamp for purposes of encrypting the second view by the second device and decrypting the second view by the third device, wherein:

the pseudo-random number generator receives as input the seed, the base timestamp and the transaction timestamp and produces as output a random number;

the transformation function takes as input the said random number and the PIE and produces as output a number; and

the hash function takes as input the said number and produces as output the encryption key.

18. The computer system as in claim 3, wherein the response from the third device to the second device comprises a plaintext part and an encrypted part, wherein the plaintext part of the response includes the device identification of the second device and a transaction timestamp and may optionally include a transaction ID, and the encrypted part includes data corresponding to a response to the agreement and may include one or more of a device identification of the second device, device identification of the first device, a transaction ID, a message digest, a transaction timestamp, the user ID of the user of the second device and padding data.

19. The computer system of claim 18, in which the encrypted part of the response of the third device to the second device is encrypted using a symmetric secret encryption key and a cryptographic algorithm in which the key and cryptographic algorithm are known only to the second device and the third device and the keys are not communicated either between the first device and the second device or the second device and the third device.

20. The computer system of claim 19 in which the encryption key used by the third device to encrypt the encrypted part of the response is generated based on a secret value comprising a Private Identification Entry (PIE), and a deterministic algorithm based on a set of stored parameters and functions, wherein the PIE is known only to the operator of the second device and is not stored on the second device and is also known to the third device and the deterministic algorithm and the set of stored parameters and functions are known only to the second device and the third device.

21. The computer system of claim 20 in which each of the third device and the second device comprises a known pseudo-random number generator, a known hash function and a known transformation function and deterministically produce identical keys given the same transaction timestamp for purposes of encrypting the response by the third device and decrypting the response by the second device, wherein:

the pseudo-random number generator receives as input the seed, the base timestamp and the transaction timestamp and produces as output a random number;

the transformation function takes as input the said random number and the PIE and produces as output a number; and

the hash function takes as input the said number and produces as output the encryption key.

22. A computer system securely transmitting and verifying a multi-party agreement among N parties where N is larger than or equal to two, said computer system comprising:

a collection of N devices, each device operated by a party to the agreement, developing and transmitting its view

of the multi-party agreement; where each view includes an encrypted part and an unencrypted, plaintext part, and

an N+1-st device comprising a verification party, receiving the views from the N agreement parties, decrypting the encrypted part of each view, affirming that each view was developed by its associated agreement party, comparing the decrypted views from the N agreement parties and transmitting a response signal based on the outcome of the comparison.

23. The computer system as in claim 22, wherein either each of the N devices independently transmits its view to the N+1-st device comprising the verification party, or, each of the N devices concatenates its view to a list of views until all N views are collected so that each view appears once in the list and the list is then transmitted to the N+1-st device.

24. The computer system as in claim 23, wherein the N+1-st device transmits a response to each of the N devices in the multi-party agreement either if it decides that the decrypted part of all N views from the N devices in the multi-party agreement match each other for the purposes of the agreement or if it decides that the decrypted part of all N views from the N devices in the multi-party agreement do not match each other for the purposes of the agreement.

25. The computer system as in claim 24, wherein the plaintext part of the view of a device includes the device identification and the transaction timestamp of each device, and the encrypted part includes data corresponding to a multi-party agreement and the device identification for each of the other devices in the multi-party agreement.

26. The computer system as in claim 25, wherein the plaintext portion of a view from a device may optionally include a transaction ID, the number of parties in the agreement and the encrypted portion may include one or more of a transaction ID, the number of parties in the agreement, a message digest, a transaction timestamp, the user ID of the user of the device and padding data.

27. The computer system of claim 26, in which the encrypted part of the view from each device is encrypted using a symmetric secret encryption key, and a cryptographic algorithm in which the key and cryptographic algorithm are known only to that device and the N+1-st device and the keys are not communicated.

28. The computer system of claim 27 in which the encryption key used by a device in the multi-party agreement to encrypt the view is generated based on a secret value comprising a Private Identification Entry (PIE), and a deterministic algorithm based on a set of stored parameters and functions, wherein the PIE is known only to the operator of a device and is not stored on the device and is also known to the N+1-st device that receives its encrypted view, and the deterministic algorithm and the set of stored parameters and functions are known only to a device and the N+1-st device.

29. The computer system of claim 28 in which a device in the multi-party agreement and the N+1-st device comprises a known pseudo-random number generator, a known hash function and a known transformation function and deterministically produce identical encryption keys given the same transaction timestamp, for purposes of encrypting the view by the device and decrypting the view by the N+1-st device, wherein:

the pseudo-random number generator receives as input the seed, the base timestamp and the transaction timestamp and produces as output a random number;

the transformation function takes as input the said random number and the PIE and produces as output a number; and

the hash function takes as input the said number and produces as output the encryption key.

30. The computer system as in claim 29, wherein the response from the N+1-st device to a device in the multi-party agreement comprises of a plaintext part and an encrypted part, wherein the plaintext part of the response includes the device identification of the device and a transaction timestamp and may optionally include a transaction ID, and the encrypted part includes data corresponding to a response to the agreement and may include one or more of a device identification of the device, device identification of the devices of the parties in the agreement, a transaction ID, a number of parties in the agreement, a message digest, a transaction timestamp, the user ID of the user of the device and padding data.

31. The computer system of claim 30, in which the encrypted part of the response of the N+1-st device to a device in the multi-party agreement is encrypted using a symmetric secret encryption key and a cryptographic algorithm in which the key and cryptographic algorithm are known only to the device and the N+1-st device and the keys are not communicated.

32. The computer system of claim 31 in which the encryption key used by the N+1-st device to encrypt the response to a device in the multi-party agreement is generated based on a secret value comprising a Private Identification Entry (PIE), and a deterministic algorithm based on a set of stored parameters and functions, wherein the PIE is known only to the operator of the device and is not stored on the device and is also known to the N+1-st device and the deterministic algorithm and the set of stored parameters and functions are known only to the device and the N+1-st device.

33. The computer system of claim 32 in which a device in the multi-party agreement and the N+1-st device comprises a known pseudo-random number generator, a known hash function and a known transformation function and deterministically produce identical keys given the same transaction timestamp for purposes of encrypting the response by the N+1-st device and decrypting the response by said device in the multi-party agreement, wherein:

the pseudo-random number generator receives as input the seed, the base timestamp and the transaction timestamp and produces as output a random number;

the transformation function takes as input the said random number and the PIE and produces as output a number; and

the hash function takes as input the said number and produces as output the encryption key.

34. A method of a computer system securely transmitting and verifying a two-party agreement, said method comprising:

developing and transmitting, by a first device operated by the first party, a first view of the two-party agreement, the first view including an encrypted part and an unencrypted plaintext part;

developing and transmitting, by a second device operated by the second party, a second view of the two-party agreement, the second view including an encrypted part and an unencrypted plaintext part; and

receiving, by a third device comprising a verification party, the first view and the second view, decrypting the encrypted part of the first view and the encrypted part of the second view, affirming that the first view was developed by the first party and that the second view was developed by the second party, comparing the decrypted part of the first view with the decrypted part of the second view, and transmitting a signal based on the outcome of the comparison.

35. A method of a computer system securely transmitting and verifying a multi-party agreement among N parties where N is larger than or equal to two, said method comprising:

developing and transmitting, by a collection of N devices each device operated by a party to the agreement, its view of the multi-party agreement; where each view includes an encrypted part and an unencrypted, plaintext part, and

receiving, by an N+1-st device comprising a verification party, the views from the N agreement parties, decrypting the encrypted part of each view, affirming that each view was developed by its associated agreement party, comparing the decrypted views from the N agreement parties and transmitting a response signal based on the outcome of the comparison.

36. A computer readable storage controlling a computer to securely transmit and verify a two-party agreement, by the functions comprising:

developing and transmitting, by a first device operated by the first party, a first view of the two-party agreement, the first view including an encrypted part and an unencrypted plaintext part;

developing and transmitting, by a second device operated by the second party, a second view of the two-party agreement, the second view including an encrypted part and an unencrypted plaintext part; and

receiving, by a third device comprising a verification party, the first view and the second view, decrypting the encrypted part of the first view and the encrypted part of the second view, affirming that the first view was developed by the first party and that the second view was developed by the second party, comparing the decrypted part of the first view with the decrypted part of the second view, and transmitting a signal based on the outcome of the comparison.

37. A computer readable storage controlling a computer to securely transmit and verify a multi-party agreement among N parties where N is larger than or equal to two, by the functions comprising:

developing and transmitting, by a collection of N devices each device operated by a party to the agreement, its view of the multi-party agreement; where each view includes an encrypted part and an unencrypted, plaintext part, and

receiving, by an N+1-st device comprising a verification party, the views from the N agreement parties, decrypting the encrypted part of each view, affirming that each view was developed by its associated agreement party, comparing the decrypted views from the N agreement parties and transmitting a response signal based on the outcome of the comparison.

* * * * *