US009270649B1

(12) **United States Patent**
Ng

(10) **Patent No.: US 9,270,649 B1**
(45) **Date of Patent: Feb. 23, 2016**

(54) **SECURE SOFTWARE AUTHENTICATOR DATA TRANSFER BETWEEN PROCESSING DEVICES**

(71) Applicant: **EMC Corporation**, Hopkinton, MA (US)

(72) Inventor: **Millie K. Ng**, Bedford, MA (US)

(73) Assignee: **EMC Corporation**, Hopkinton, MA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 164 days.

(21) Appl. No.: **13/793,327**

(22) Filed: **Mar. 11, 2013**

(51) **Int. Cl.**
H04L 29/06 (2006.01)

(52) **U.S. Cl.**
CPC ***H04L 63/0428*** (2013.01)

(58) **Field of Classification Search**
CPC G06Q 30/06; G06Q 20/3229; G06Q 20/35785; G06Q 20/353; G06Q 20/3823; G06Q 20/10; G06Q 20/105; G06Q 20/108; G06Q 20/40; G06Q 20/3821; H04L 63/105; H04L 63/0853; H04L 63/0869; H04L 63/0838; H04L 63/0846; H04L 63/16; H04L 63/205; H04L 67/04; H04L 67/14; H04L 9/0869; H04L 9/0863; H04W 12/06; H04W 12/08; H04W 56/002; H04B 5/00; H04B 5/02; H04B 5/0031; H04B 5/0062
USPC 713/181
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,800,590	A *	1/1989	Vaughan	713/184
6,237,095	B1 *	5/2001	Curry	G06Q 20/06 713/178
6,985,583	B1 *	1/2006	Brainard et al.	380/44
8,219,817	B2 *	7/2012	Filreis	H04L 63/123 713/176
8,233,841	B2 *	7/2012	Griffin et al.	455/41.1
8,307,410	B2 *	11/2012	Martin et al.	726/4
8,327,429	B2 *	12/2012	Speyer et al.	726/9

(Continued)

OTHER PUBLICATIONS

Securology, 'Soft tokens aren't tokens at all', Creative Commons, Nov. 20, 2007, entire document, http://securology.blogspot.com/2007/11/soft-tokens-arent-tokens-at-all.html.*

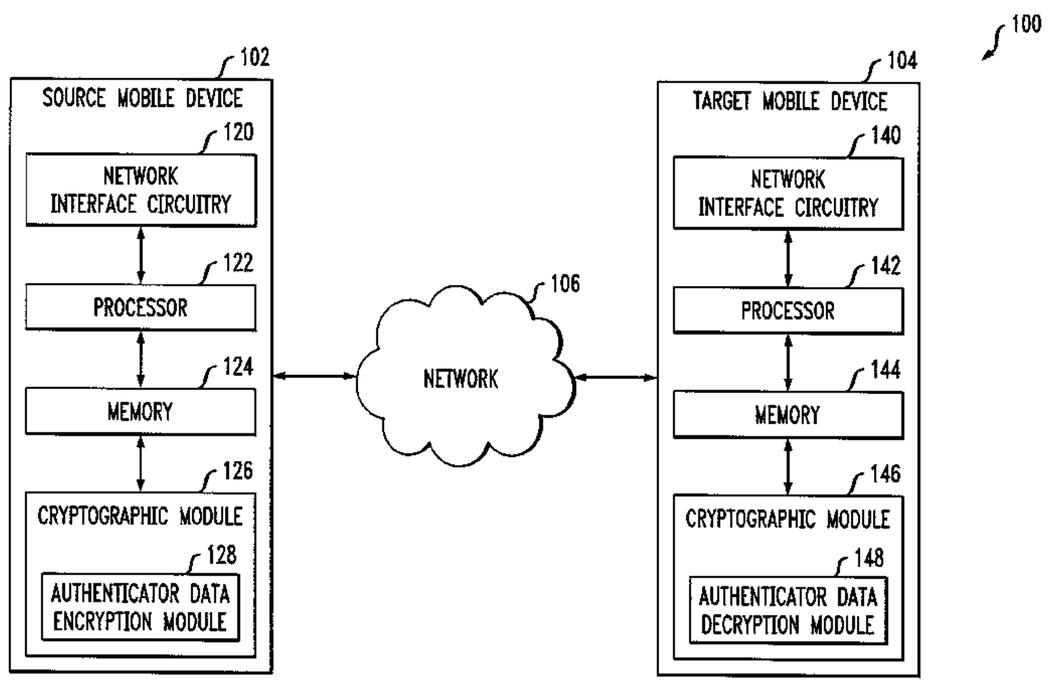
(Continued)

Primary Examiner — Christopher Brown
Assistant Examiner — Ronald Baum
(74) *Attorney, Agent, or Firm* — Ryan, Mason & Lewis, LLP

(57) **ABSTRACT**

A method comprises establishing a network connection between the first processing device and the second processing device for transfer of data associated with a software authenticator from the first processing device to the second processing device, encrypting the software authenticator data with encryption that is separate from encryption used for the network connection, and transferring the encrypted software authenticator data from the first processing device to the second processing device. Another method comprises establishing the network connection between the first processing device and the second processing device for transfer of the software authenticator data, receiving encrypted data from the first processing device, wherein the encrypted data has encryption that is separate from encryption used for the network connection, decrypting the encrypted data to obtain data associated with a software authenticator and importing the software authenticator data into a software authenticator stored in a memory of the second processing device.

25 Claims, 3 Drawing Sheets



(56) **References Cited**

U.S. PATENT DOCUMENTS

8,392,702	B2 *	3/2013	Qiu et al.	713/159
8,595,810	B1 *	11/2013	Ben Ayed	726/8
8,745,710	B1 *	6/2014	Roth et al.	726/6
2001/0002485	A1 *	5/2001	Bisbee et al.	713/167
2002/0143855	A1 *	10/2002	Traversat et al.	709/202
2003/0115467	A1 *	6/2003	Aull et al.	713/173
2004/0088347	A1 *	5/2004	Yeager et al.	709/202
2004/0117623	A1 *	6/2004	Kalogridis et al.	713/165
2005/0021982	A1 *	1/2005	Popp et al.	713/184
2005/0136964	A1 *	6/2005	Le Saint et al.	455/522
2005/0160269	A1 *	7/2005	Akimoto	H04L 63/0272 713/171
2006/0059346	A1 *	3/2006	Sherman et al.	713/175
2006/0208066	A1 *	9/2006	Finn et al.	235/380
2007/0230694	A1 *	10/2007	Rose et al.	380/46
2007/0234064	A1 *	10/2007	Nihei	713/183
2008/0010449	A1 *	1/2008	Holtzman et al.	713/157
2009/0300738	A1 *	12/2009	Dewe et al.	726/6
2010/0024004	A1 *	1/2010	Boegelund et al.	726/3
2010/0199336	A1 *	8/2010	Tan	726/6
2010/0257578	A1 *	10/2010	Shukla et al.	726/1
2011/0016320	A1 *	1/2011	Bergsten et al.	713/170
2012/0124651	A1 *	5/2012	Ganesan et al.	726/4
2012/0174198	A1 *	7/2012	Gould et al.	726/6
2012/0278241	A1 *	11/2012	Brown et al.	705/67
2013/0061055	A1 *	3/2013	Schibuk	713/172
2013/0091544	A1 *	4/2013	Oberheide et al.	726/1
2014/0201536	A1 *	7/2014	Fiske	713/183

OTHER PUBLICATIONS

Y Combinator, 'RSA hit by targeted attacks, SecurID 2-factor auth possibly compromised (rsa.com)', Y Combinator (blog), Mar. 2011, entire document, https://news.ycombinator.com/item?id=2338368.*

* cited by examiner

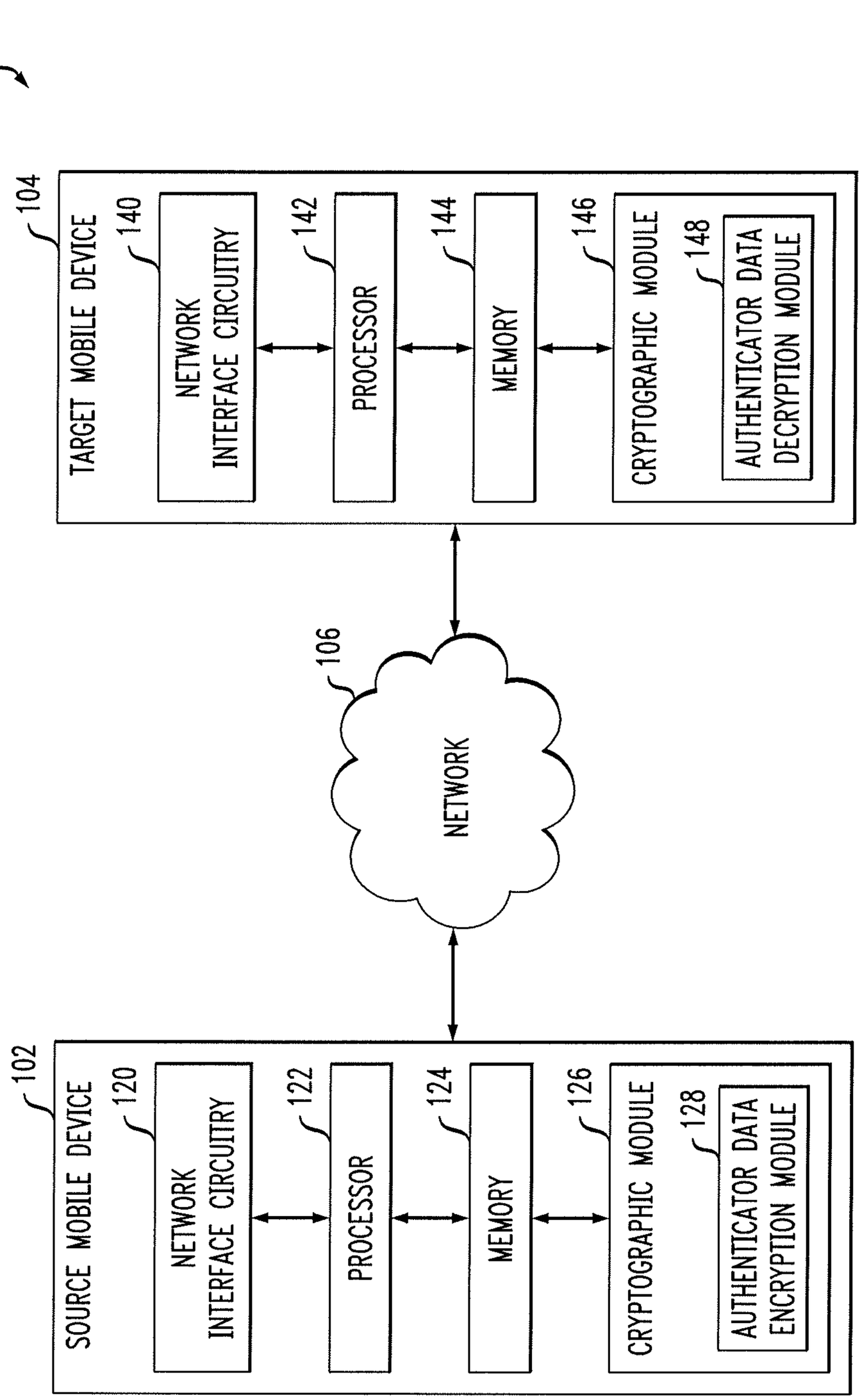

100
SOURCE MOBILE DEVICE
102
NETWORK INTERFACE CIRCUITRY
120
PROCESSOR
122
MEMORY
124
CRYPTOGRAPHIC MODULE
126
AUTHENTICATOR DATA ENCRYPTION MODULE
128
NETWORK
106
TARGET MOBILE DEVICE
104
NETWORK INTERFACE CIRCUITRY
140
PROCESSOR
142
MEMORY
144
CRYPTOGRAPHIC MODULE
146
AUTHENTICATOR DATA DECRYPTION MODULE
148

FIG. 1

FIG. 2

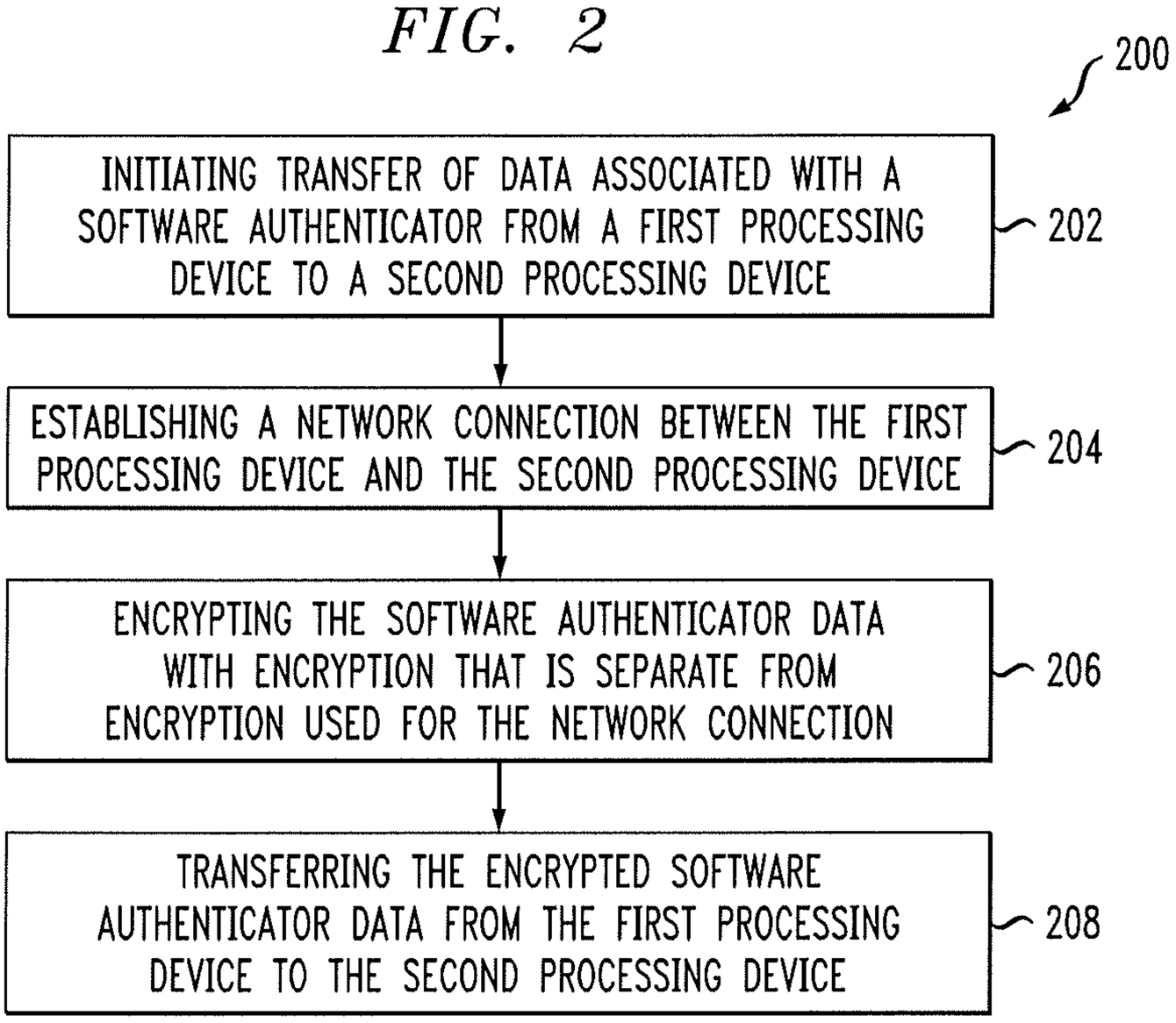

200
INITIATING TRANSFER OF DATA ASSOCIATED WITH A SOFTWARE AUTHENTICATOR FROM A FIRST PROCESSING DEVICE TO A SECOND PROCESSING DEVICE
202
ESTABLISHING A NETWORK CONNECTION BETWEEN THE FIRST PROCESSING DEVICE AND THE SECOND PROCESSING DEVICE
204
ENCRYPTING THE SOFTWARE AUTHENTICATOR DATA WITH ENCRYPTION THAT IS SEPARATE FROM ENCRYPTION USED FOR THE NETWORK CONNECTION
206
TRANSFERRING THE ENCRYPTED SOFTWARE AUTHENTICATOR DATA FROM THE FIRST PROCESSING DEVICE TO THE SECOND PROCESSING DEVICE
208

FIG. 3

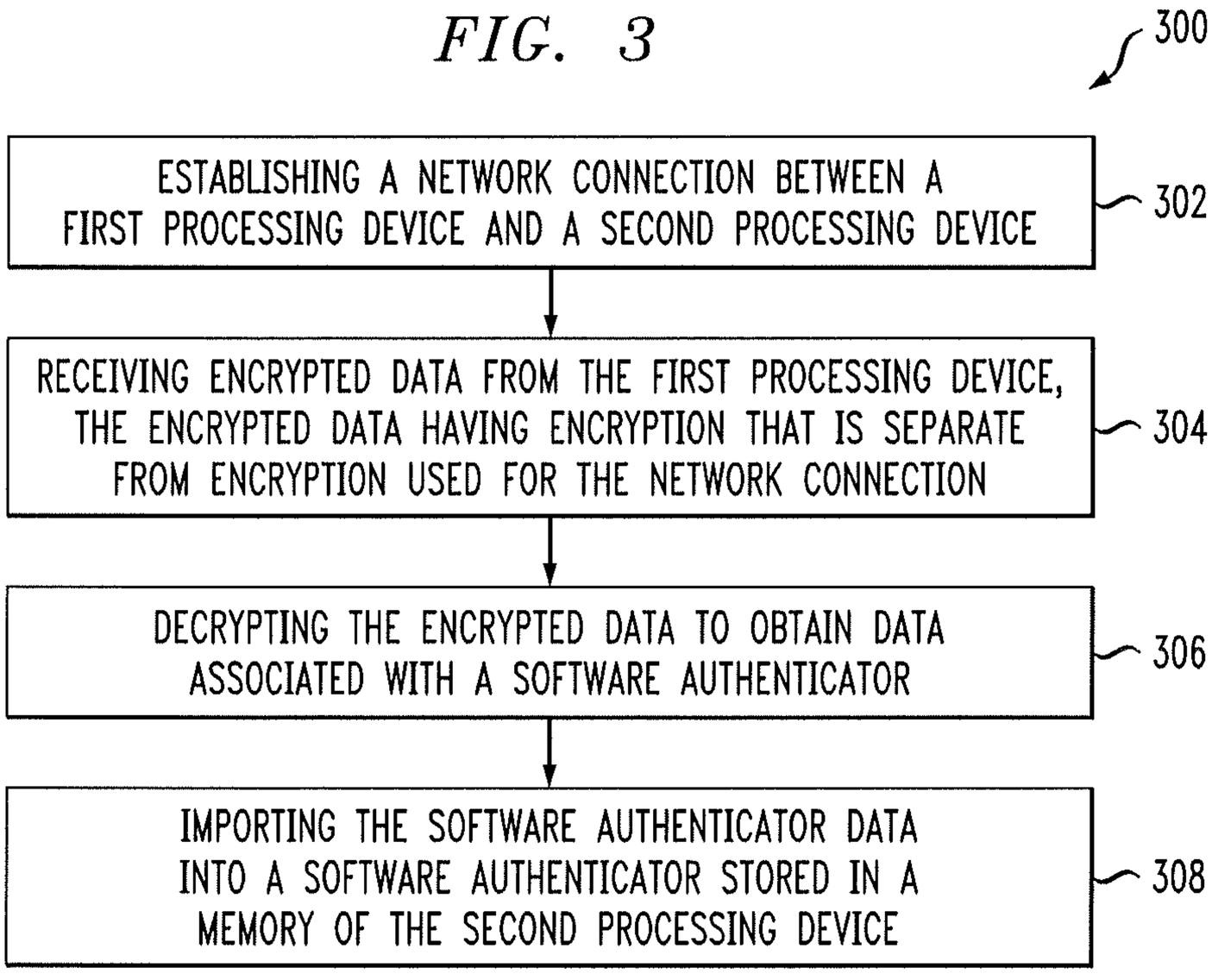
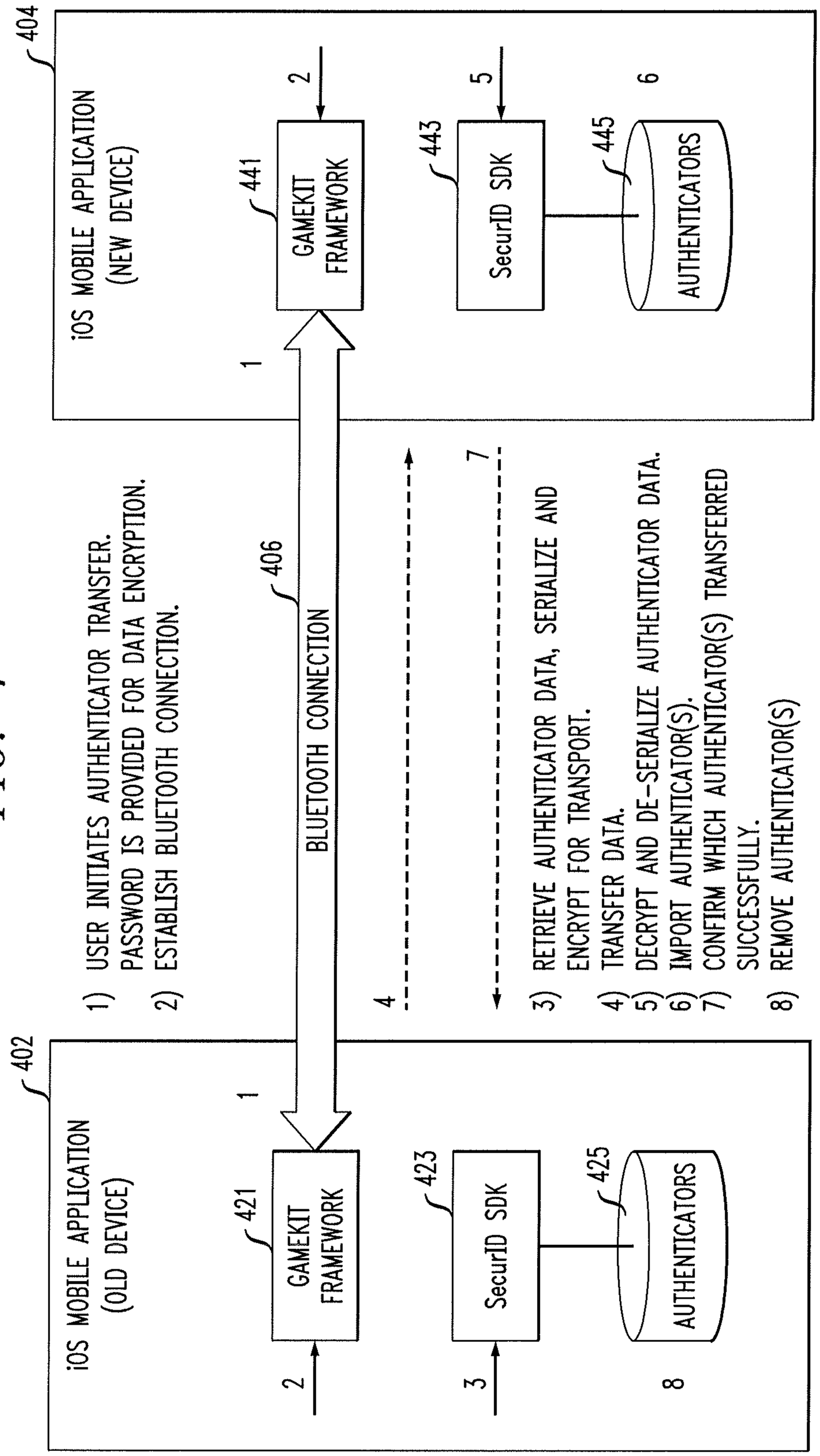

300
ESTABLISHING A NETWORK CONNECTION BETWEEN A FIRST PROCESSING DEVICE AND A SECOND PROCESSING DEVICE
302
RECEIVING ENCRYPTED DATA FROM THE FIRST PROCESSING DEVICE, THE ENCRYPTED DATA HAVING ENCRYPTION THAT IS SEPARATE FROM ENCRYPTION USED FOR THE NETWORK CONNECTION
304
DECRYPTING THE ENCRYPTED DATA TO OBTAIN DATA ASSOCIATED WITH A SOFTWARE AUTHENTICATOR
306
IMPORTING THE SOFTWARE AUTHENTICATOR DATA INTO A SOFTWARE AUTHENTICATOR STORED IN A MEMORY OF THE SECOND PROCESSING DEVICE
308

FIG. 4


404
iOS MOBILE APPLICATION
(NEW DEVICE)
2
5
6
441
GAMEKIT
FRAMEWORK
443
SecurID SDK
445
AUTHENTICATORS
1
406
BLUETOOTH CONNECTION
4
7
1) USER INITIATES AUTHENTICATOR TRANSFER.
PASSWORD IS PROVIDED FOR DATA ENCRYPTION.
2) ESTABLISH BLUETOOTH CONNECTION.
3) RETRIEVE AUTHENTICATOR DATA, SERIALIZE AND
ENCRYPT FOR TRANSPORT.
4) TRANSFER DATA.
5) DECRYPT AND DE-SERIALIZE AUTHENTICATOR DATA.
6) IMPORT AUTHENTICATOR(S).
7) CONFIRM WHICH AUTHENTICATOR(S) TRANSFERRED
SUCCESSFULLY.
8) REMOVE AUTHENTICATOR(S)
402
iOS MOBILE APPLICATION
(OLD DEVICE)
1
421
GAMEKIT
FRAMEWORK
423
SecurID SDK
425
AUTHENTICATORS
2
3
8

SECURE SOFTWARE AUTHENTICATOR DATA TRANSFER BETWEEN PROCESSING DEVICES

FIELD

The field relates generally to cryptography, and more particularly to software authenticators implemented in processing devices.

BACKGROUND

One-time passcode (OTP) authentication tokens may be implemented in hardware and software. Hardware authentication tokens are typically implemented as small, hand-held devices that display a series of passcodes over time. A user equipped with such an authentication token reads the currently displayed passcode and enters it into a computer or other element of an authentication system as part of an authentication operation. This type of dynamic passcode arrangement offers a significant security improvement over authentication based on a static password. Software authentication tokens, or software authenticators, can be implemented in the form of software installed on a processing device such as a computer, mobile phone, tablet, etc.

Conventional authentication tokens include both time-synchronous and event-synchronous tokens.

In a typical time-synchronous token, the displayed passcodes are based on a secret value and the time of day. A verifier with access to the secret value and a time of day clock can verify that a given presented passcode is valid.

One particular example of a time-synchronous authentication token is the RSA SecurID® user authentication token, commercially available from RSA, The Security Division of EMC Corporation, of Bedford, Mass., U.S.A.

Event-synchronous tokens generate passcodes in response to a designated event, such as a user pressing a button on the token. Each time the button is pressed, a new passcode is generated based on a secret value and an event counter. A verifier with access to the secret value and the current event count can verify that a given presented passcode is valid.

Other known types of authentication tokens include hybrid time-synchronous and event-synchronous tokens.

Passcodes can be communicated directly from the authentication token to a computer or other element of an authentication system, instead of being displayed to the user. For example, a wired connection such as a universal serial bus (USB) interface may be used for this purpose. Wireless authentication tokens are also known. In authentication tokens of this type, the passcodes are wirelessly communicated to a computer or other element of an authentication system. These wired or wireless arrangements, also referred to herein as connected tokens, save the user the trouble of reading the passcode from the display and manually entering it into the computer.

Hardware and software authentication tokens and other types of OTP devices are typically programmed with a random seed or other type of key that is also stored in a token record file. The record file is loaded into an authentication server, such that the server can create matching passcodes for the authentication token based on the key and the current time or current event count.

SUMMARY

Illustrative embodiments of the present invention provide techniques for secure transfer of software authenticator data between processing devices.

In one embodiment, a method comprises establishing a network connection between the first processing device and the second processing device for transfer of data associated with a software authenticator from the first processing device to the second processing device, encrypting the software authenticator data with encryption that is separate from encryption used for the network connection and transferring the encrypted software authenticator data from the first processing device to the second processing device.

In another embodiment, a method comprises establishing a network connection between a first processing device and a second processing device for transfer of data associated with a software authenticator from the first processing device to the second processing device, receiving encrypted data from the first processing device, wherein the encrypted data has encryption that is separate from encryption used for the network connection, decrypting the encrypted data to obtain data associated with a software authenticator and importing the software authenticator data into a software authenticator stored in a memory of the second processing device.

These and other features and advantages of embodiments of the present invention will become more readily apparent from the accompanying drawings and the following detailed description.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a simplified block diagram of an exemplary communication system in which embodiments of the present invention may be implemented.

FIG. **2** illustrates a methodology for transfer of software authenticator data, according to an embodiment of the invention.

FIG. **3** illustrates a methodology for importing software authenticator data, according to an embodiment of the invention.

FIG. **4** illustrates an example of software authenticator transfer, according to an embodiment of the invention.

DETAILED DESCRIPTION

Illustrative embodiments of the present invention will be described herein with reference to exemplary communication systems and associated processing devices, networks, servers, etc. It is to be appreciated, however, that the invention is not restricted to use with the particular illustrative system and device configurations shown. Accordingly, the term "communication system" as used herein is intended to be broadly construed, so as to encompass, for example, systems in which multiple processing devices communicate with one another over a network.

The term "passcode" as used herein is intended to include authentication information such as OTPs, or more generally any other information that may be utilized for cryptographic authentication purposes. Although the illustrative embodiments will be described below primarily in the context of OTPs, it is to be appreciated that the invention is more broadly applicable to any other type of passcode.

As will be described, the present invention in one or more illustrative embodiments provides techniques for facilitating secure transfer of software authenticators between processing devices.

Today, users often replace processing devices such as mobile phones, tablets, laptops, etc. frequently. New devices are released every few months, and some users will upgrade their devices to remain at the cutting edge. In addition, typical subscription plans for mobile phones are 2-year plans, where

users are eligible for a free or discounted upgrade at or around the end of the 2-year plan. Thus, even users who do not upgrade devices every few months or every year will typically upgrade devices such as mobile phones every two years. Thus, many users will end up replacing devices before software authenticators expire. For example, the RSA SecurID® authenticator typically has a lifetime of 10 years.

Software authenticators may employ mechanisms which ensure that a given user's authenticator is installed and running on the device it is intended for. For example, a user may register their mobile phone for use with a particular software authenticator. To prevent security breaches, software authenticators such as the RSA SecurID® authenticator will ensure that the authenticator is installed on the correct registered device.

In conventional devices, transfer of a software authenticator to a new device is complicated and may be vulnerable to attack. OTP generation in software authenticators typically requires an application which includes a seed, a serial number, a display interval, display digits, and other meta-data. Software authenticators can be distributed to end-users by an authenticator management server administrator using one of a number of methods. In one method, the software authenticator is distributed using a certificate file such as the SDTID file used for RSA SecurID® Software Authenticators. In other methods, the software authenticator may be distributed using a Compressed Token Format (CTF) string or dynamic seed provisioning. An example of dynamic seed provisioning is the Cryptographic Token Key Initialization Protocol (CT-KIP).

To transfer a software authenticator, a user must contact an information technology (IT) help desk, authenticator management server administrator or some other entity associated with the software authenticator to re-issue the software authenticator to a new device or to issue a new authenticator for the new device. By way of example, a user wishing to transfer a software authenticator may be forced to perform a number of tasks. A user may request a transfer by placing a support call to an IT help desk. The user may then be required to install a software authenticator application on the new device and email a device binding identification to the IT help desk. The authenticator or authenticator data can be bound to the device identification and e-mailed or otherwise sent to the end-user. The e-mail or other communication may embody the authenticator or contain links which initiate dynamic seed provisioning for the authenticator. The end-user follows the instructions in the e-mail or other communication to import and provision or re-provision the software authenticator.

Such techniques for transferring and re-provisioning software authenticators can have significant drawbacks. For example, the IT help desk or other support resource may have limited hours, or have significant wait times. Thus, a user may be inconvenienced when attempting to transfer and re-provision a software authenticator. Such techniques may also require significant costs for the software authenticator issuer. In addition, such techniques may expose sensitive and secure data to attack. For example, if provisioning of authenticators is done via e-mail, the e-mail may contain all of the data pertaining to a user's authenticator. Thus, if a user's e-mail account becomes compromised, the software authenticator may also be compromised. A user may also forward the e-mail to others, running the risk of releasing data which should not be exposed to third parties.

Accordingly, embodiments of the invention provide techniques which give end-users a self-service option for transferring software authenticators that is more convenient, less expensive and simpler than conventional transfer and re-provisioning techniques. Embodiments of the invention provide methods for end-users to directly transfer authenticator data in a secure manner from one device to another wirelessly.

While embodiments of the invention described below may refer to a software authenticator which comprises a time-synchronous token such as the RSA SecurID® token, it is important to note that embodiments of the invention are not limited solely to use with time-synchronous tokens. Instead, embodiments of invention may be utilized with event-synchronous authentication tokens, challenge-response tokens, hash-chain tokens, or hybrid tokens that incorporate multiple such capabilities, such as hybrid time-synchronous and event-synchronous tokens. A given software authentication token may be a connected token or a disconnected token, or one capable of operating in both connected and disconnected modes.

In addition, while various embodiments are described below with respect to transfer of a single software authenticator, embodiments of the invention may be utilized to transfer multiple software authenticators, either serially or in parallel with one another.

FIG. **1** shows a communication system **100** comprising a source mobile device **102** and a target mobile device **104** connected via a network **106**. While FIG. **1** shows source and target mobile devices **102** and **104**, respectively, embodiments of the invention are not limited solely to use with mobile devices. Instead, embodiments of the invention may be used more generally with processing devices, which include mobile devices such as cell phones, tablets, laptops, personal digital assistants (PDAs), etc. as well as other computing and communication devices.

The source mobile device **102** comprises network interface circuitry **120**, a processor **122**, a memory **124** and a cryptographic module **126** comprising an authenticator data encryption module **128**. The target mobile device **104** comprises network interface circuitry **140**, a processor **142**, a memory **144** and a cryptographic module **146** comprising an authenticator data decryption module **148**.

The processors **122** and **142** may comprise microprocessors, microcontrollers, application-specific integrated circuits (ASICs), field-programmable gate arrays (FPGAs) or other types of processing circuitry, as well as portions or combinations of such circuitry elements.

Each of the memories **124** and **144** may comprise random access memory (RAM), read-only memory (ROM), a hard disk drive (HDD), flash memory or other types of memory, in any combination. The memories **124** and **144** may be viewed as examples of what are more generally referred to herein as "computer program products" storing executable program code.

The network interface circuitries **120** and **140** allow the source mobile device **102** and the target mobile device **104**, respectively, to communicate over the network **106** with one another and with other devices, servers, etc. not shown in FIG. **1**.

The source mobile device **102** implements a cryptographic module **126** comprising an authenticator data encryption module **128**. As will be described in further detail below, the cryptographic module **126** may embody a software authenticator application, token or software authenticator instance. The authenticator data encryption module **128** allows for encrypting authenticator data required to transfer and re-provision a software authenticator.

The target mobile device **104** implements a cryptographic module **146** comprising an authenticator data decryption module **148**. As will be described in further detail below, the cryptographic module **146** may embody a software authenti-

cator application, token or software authenticator instance. The authenticator data decryption module **148** allows for decrypting authenticator data received in a software authenticator transfer used for re-provisioning the software authenticator.

While FIG. **1** shows source mobile device **102** and target mobile device **104** with authenticator data encryption module **128** and authenticator data decryption module **148**, respectively, it is important to note that in some embodiments a given device may contain both an authenticator data encryption module and an authenticator data decryption module. For example, a given device may act as a source of a software authenticator in one instance and a target in another instance. Thus, the designation of a device as a source or target device in FIG. **1** and throughout this description is for clarity of illustration. Such designations should not be construed as limiting a particular device as being solely a source or solely a target device.

The source mobile device **102** and the target mobile device **104** may include additional components not specifically illustrated in FIG. **1** which are of a type commonly used in processing devices, as will be appreciated by those skilled in the art.

For security reasons, the network **106** may be a short range or private network type. For example, many processing devices including cellular phones come equipped with Bluetooth network interface circuitry. As another example, near field communication (NFC) network interface circuitry is equipped in many newer processing devices such as cellular phones. The network **106**, however, may be another network type such as a WiFi or WiMAX network, a cellular network, a telephone or cable network, a local area network (LAN), a wide area network (WAN) a global computer network such as the Internet, or various portions or combinations of these and other types of networks.

It is to be appreciated that the particular set of elements shown in FIG. **1** in system **100** is presented by way of example, and in other embodiments additional or alternative elements may be used. Thus, other embodiments may include additional networks and additional devices or servers.

As mentioned previously, various elements of system **100** such as the source mobile device **102** and the target mobile device **104**, their associated functional modules such as the cryptographic modules **126** and **146**, respectively, and other elements may be implemented at least in part in the form of software. Such software is stored and executed utilizing respective memory and processor elements of at least one processing device. The system **100** may include additional or alternative processing platforms, as well as numerous distinct processing platforms in any combination, with each such platform comprising one or more computers, servers, storage devices or other types of processing devices.

Such processing platforms may include cloud infrastructure comprising virtual machines (VMs) and one or more associated hypervisors. An example of a commercially available hypervisor platform that may be used to implement portions of the communication system **100** is the VMware® vSphere™ which may have an associated virtual infrastructure management system such as the VMware® vCenter™. The underlying physical machines may comprise one or more distributed processing platforms that include storage products, such as VNX and Symmetrix VMAX, both commercially available from EMC Corporation of Hopkinton, Mass. A variety of other storage products may be utilized to implement at least a portion of the system **100**.

FIGS. **2-3** illustrate methodologies for transferring software authenticator data. FIG. **2** illustrates a methodology **200** for transfer of software authenticator data from the perspective of a first processing device, an example of which is the source mobile device **102** in FIG. **1**. The methodology **200** begins with initiating **202** transfer of data associated with a software authenticator from a first processing device to a second processing device. The second processing device may be the target mobile device **104** in FIG. **1**.

As described above, a software authenticator may include various data such as a seed, a serial number, etc. In some instances, a software authenticator application or token may be installed on both the first processing device and the second processing device, while only the first processing device has the data such as the seed and serial number required for the software authenticator to operate. In such instances, transfer of the software authenticator only requires transfer of the seed, serial number or other data associated with the software authenticator without transferring an entire software authenticator application or token. In other embodiments, the second processing device may receive the entire software authenticator application or token from the first processing device.

The methodology **200** continues with establishing **204** a network connection between the first processing device and the second processing device. As discussed above, for improved security some embodiments of the invention may establish a short range or private network connection such as a Bluetooth or NFC connection between the first and second processing devices. In other embodiments, various other network connections, including combinations of network connections, may be established.

The software authenticator data is encrypted **206** with encryption that is separate from encryption used for the network connection. The authenticator data encryption module **128** may be used to encrypt **206** the software authenticator data. Many network types provide for some type or layer of encryption to be applied to data which is transmitted over the network. For example, Bluetooth connections use shared-key authentication, strong encryption algorithms and can operate in various security modes. However, Bluetooth and other wireless network connections can introduce security vulnerabilities depending on how they are implemented and used in particular devices. This can lead to a compromise of device or software authenticator data, or render the transfer of software authenticator data susceptible to eavesdropping which is a serious concern when sensitive data such as the software authenticator data is being transmitted between devices.

To ensure secure transfer of the software authenticator data, embodiments of the invention encrypt the software authenticator data with encryption that is separate from the encryption used for the network connection. In some embodiments, portions of the authenticator data may also be separately encrypted. For example, if transfer of a given software authenticator requires transfer of both a seed value and a serial number, the seed value and the serial number may be encrypted with encryption separate from one another and separate from encryption used for the network connection. The encrypted software authenticator data is transferred **208** from the first processing device to the second processing device.

The first and second processing devices may exchange a binding identification to ensure that the software authenticator data is installed only on the correct or authorized device. In some embodiments, this binding identification can be used in the encryption of the software authenticator data or may be encrypted along with the software authenticator data, or otherwise folded into the encryption used for the software authenticator data.

The encryption used to encrypt that software authenticator data may be implemented using a number of conventional techniques and processes, such as those disclosed in A. J. Menezes et al., Handbook of Applied Cryptography, CRC Press, 1997, which is incorporated by reference herein. These conventional processes, being well known to those skilled in the art, will not be described in further detail herein, although embodiments of the invention may incorporate aspects of such processes.

FIG. **3** illustrates a methodology **300** for importing and provisioning a software authenticator from the perspective of the second processing device referred to in FIG. **2**. The target mobile device **104** in FIG. **1** is an example of the second processing device referred to in FIG. **3**. The methodology **300** begins with establishing **302** a network connection between the first processing device and the second processing device.

It is important to note that although methodologies **200** and **300** describe the first processing device initiating the transfer of the software authenticator data, embodiments of the invention are not limited solely to this arrangement. For example, in some embodiments the second processing device may initiate the transfer of the software authenticator data, or the first and second processing devices may both initiate a transfer of the software authenticator data. In addition, while embodiments are described with respect to a network connection established for the transfer of software authenticator data, the network connection need not be used solely for the transfer of software authenticator data. In addition, establishing a network connection for the transfer of software authenticator data between the first and second processing devices does not require setting up a new network connection. Instead, embodiments of the invention may use one or more existing network connections between the first and second processing devices for transfer of the software authenticator data.

Methodology **300** continues with receiving **304** encrypted data from the first processing device. As described above, this encrypted data has encryption that is separate from encryption used for the network connection. The encrypted data is decrypted **306** to obtain the software authenticator data. The authenticator data decryption module **148** may be used to perform the decryption **306**. The software authenticator data is then imported **308** or provisioned into a software authenticator stored in a memory of the second processing device. As discussed above, in some embodiments the software authenticator data may comprise a seed value, the seed value and a serial number, or some other data associated with a software authenticator. The software authenticator data may also comprise the software authenticator application itself. The memory of the second processing device may be the memory **144** in FIG. **1** or may be another memory such as a memory in the cryptographic module **146**.

FIG. **4** illustrates a methodology for software authenticator exchange between two Apple® devices running the iOS mobile operating system. Old and new devices run respective iOS applications **402** and **404**. The old device is an example of a source device, while the new device is an example of a target device. However, as discussed above, a given device may be a source device in one instance and a target device in another instance. Thus, a given device may be configured to both transfer and receive software authenticators in accordance with embodiments of the invention.

In addition, although FIG. **4** illustrates a methodology for transfer of a software authenticator between two Apple® devices running the iOS mobile operating system, embodiments are not limited solely for use with devices miming the iOS mobile operating system. For example, devices running the Google® Android™ platform may also be used. In addition, software authenticators may be exchanged between a device running the iOS mobile operating system and a device running the Android™ platform, or between devices running various other operating systems. In addition, while the methodology in FIG. **4** will be described with respect to transfer of software authenticator data over a Bluetooth connection, embodiments are not limited solely to use with Bluetooth network connections. Instead, as detailed above, a variety of network connections may be used, including NFC, WiFi, infrared, etc.

FIG. **4** begins with step **1**, where users of the old and new devices initiate authenticator transfer in their respective iOS mobile applications **402** and **404**. In step **2**, the respective iOS mobile applications **402** and **404** use respective Gamekit Frameworks **421** and **441** to establish a Bluetooth connection **406**. It is important to note that while FIG. **4** shows the iOS mobile applications **402** and **404** utilizing the Gamekit Frameworks **421** and **441** to establish the Bluetooth connection, one or both of the iOS mobile applications **402** and **404** may alternatively use the iOS Core Bluetooth Framework. For devices utilizing the Android™ platform, the package android.bluetooth may be used for managing Bluetooth functionality. One skilled in the art will readily appreciate that the specific frameworks and/or packages used to implement the Bluetooth or other network connection may vary depending on the operating system version running on the devices.

In step **3**, the iOS mobile application **402** retrieves authenticator data from the SecurID® software development kit (SDK) **423** for RSA SecurID® authentication tokens, serializes the software authenticator data and encrypts it for transport. The SecurID® SDK **423** is suitably modified for implementing embodiments of the invention. It is important to note, however, that other SDKs, tools and information may be used for other types of software authenticators. The SecurID® SDK **423** is modified to be capable of serializing the software authenticator data. In some embodiments of the invention, the software authenticator data is serialized to JavaScript Object Notation (JSON). The software authenticator data may be encrypted using a key derived from a user-specified password or custom key derivation function (KDF). The iOS mobile application **402** may require the user to specify a strong password for encryption and decryption of the software authenticator data. Other encryption methods and algorithms may also be used for encrypting the software authenticator data in various other embodiments of the invention.

The software authenticator data is transferred from the old device to the new device in step **4**. Next, the iOS mobile application **404** decrypts and de-serializes the received software authenticator data, again using a suitably modified version of a SecurID® SDK **443** in step **5**. The iOS mobile application **404** uses the decrypted software authenticator data to import or provision a software authenticator into an authenticator memory **445** in step **6**. The iOS mobile application **404** will then confirm that the software authenticator was successfully transferred in step **7**. It is important to note that more than one software authenticator may be transferred between devices. As such, the confirmation in step **7** can specify a particular software authenticator or authenticators which were successfully transferred.

On receipt of the confirmation from the iOS mobile application **404**, the iOS mobile application **402** running on the old device will remove the authenticators which were successfully transferred from the authenticator memory **425** in step **8**. Various processes and protocols may be used in the iOS mobile application **402** running on the old device, and more generally the source processing devices described herein, for

removing authenticators after successful transfer of the software authenticator or authenticators. This eliminates multiple copies of the software authenticator or authenticators on different devices.

In some embodiments, once a software authenticator is successfully transferred and imported to the target processing device, the software authenticator may be re-seeded using a specific key derivation algorithm known by the target processing device and an authentication manager. This re-seeding may comprise application of a silent alarm function in a software authenticator, such that the re-seeding event is seamless to the end-user. After the software authenticator is transferred to the target processing device, an end-user will attempt to authenticate to some entity using the software authenticator. As an example, the software authenticator may generate an OTP using the software authenticator as usual. In the case of a time-based software authenticator token, the authentication manager will perform its normal OTP time-based matching. Since the software authenticator is re-seeded, this will fail and a silent alarm is triggered.

The authentication manager, in response to the silent alarm, attempts to locate an OTP match with a derived seed. Assuming the software authenticator was successfully transferred and has not been tampered with or otherwise compromised, the authentication manager will find the derived seed which matches the re-seeded software authenticator and associate the new seed with the particular software authenticator. In some embodiments, this may involve caching the new seed. As a result, the source processing device and/or any other device which has an old copy of the software authenticator is rendered useless since these devices have not been re-seeded.

In other embodiments, approaches other than the silent alarm/re-seeding protocol described above may be used. For example, in some embodiments the software authenticator may be configured to allow the end-user to register new devices. For example, the software authenticator on a source processing device, a target processing device, or some other processing device may be configured to allow an end-user to register the target processing device before, during or after transfer of the software authenticator data. The target processing device may be registered with an authenticator management server, an IT help desk, or other entity associated with the software authenticator. Registering the target processing device can cause such entities to generate a special code, key, command or other instruction which is sent to the target processing device. This special code, key, command or other instruction can cause the software authenticator on the target processing device to re-seed, thus rendering old copies of the software authenticator useless. Use of a special code, key, command or other instruction can minimize exposure of the authenticator data in comparison to techniques wherein the software authenticator data is sent in an e-mail from an IT help desk, an SDTID file, etc.

In various embodiments of the invention described above, techniques are described which allow a user to transfer software authenticators in a manner which does not require the user to contact an IT help desk, an authenticator management server administrator or other entity associated with the software authenticator. As such, embodiments of the invention permit end-users to self-service their software authenticators in a seamless manner. Embodiments of the invention, however, are not limited to arrangements wherein the user cannot contact an IT help desk, authenticator management server administrator or other entity associated with the software authenticator. Instead, in some embodiments of the invention processing devices may communicate with such entities as part of the software authenticator transfer process.

The particular processing operations and other system functionality described in conjunction with the flow diagrams of FIGS. **2-4** are presented by way of illustrative example only, and should not be construed as limiting the scope of the invention in any way. Alternative embodiments can use other types of processing operations for establishing a network connection, encrypting software authenticator data, etc. For example, the ordering of the process steps may be varied in other embodiments, or certain steps may be performed concurrently with one another rather than serially.

It is to be appreciated that the software authenticator transfer functionality such as that described in conjunction with the flow diagrams of FIGS. **2-4** and the associated examples above can be implemented at least in part in the form of one or more software programs stored in memory and executed by a processor of a processing device such as a computer or server. As mentioned previously, a memory or other storage device having such program code embodied therein is an example of what is more generally referred to herein as a "computer program product."

The foregoing examples are intended to illustrate aspects of certain embodiments of the present invention and should not be viewed as limiting in any way. Other embodiments can be configured that utilize different software authenticator transfer techniques.

It should again be emphasized that the above-described embodiments of the invention are presented for purposes of illustration only. Many variations and other alternative embodiments may be used. For example, the techniques are applicable to a wide variety of other types of processing devices and software authenticators.

As another example, in some embodiments of the invention, the communication system **100** in FIG. **1** may include multiple instances of the source mobile device **102** or the target mobile device **104**. In some embodiments, a single source mobile device can transfer one or more software authenticators to two or more target devices. A single source mobile device may also transfer one or more software authenticators to one target mobile device and one or more other software authenticators to another target mobile device. A given target mobile device may receive software authenticators from two or more source mobile devices, or receive parts of the software authenticator data associated with a given software authenticator from two or more source devices. Various other arrangements of source and target devices may be utilized.

Also, the particular configuration of communication system and processing device elements shown in FIGS. **1** and **4**, and the software authenticator transfer operations shown in FIGS. **2-4**, can be varied in other embodiments. Moreover, the various simplifying assumptions made above in the course of describing the illustrative embodiments should also be viewed as exemplary rather than as requirements or limitations of the invention. Numerous other alternative embodiments within the scope of the appended claims will be readily apparent to those skilled in the art.

What is claimed is:

1. A method comprising:

establishing a network connection between a first processing device and a second processing device for transfer of software authenticator data from the first processing device to the second processing device, the software authenticator data comprising a seed value utilized by a first software authenticator provisioned on the first processing device to generate one or more passcodes;

encrypting the software authenticator data;

transferring the encrypted software authenticator data from the first processing device to the second processing device, the software authenticator data being configured to provision a second software authenticator on the second processing device;

initiating re-seeding of the second software authenticator responsive to a successful provisioning of the second software authenticator on the second processing device;

receiving, at the first processing device from the second processing device, a confirmation indicating a successful transfer of the software authenticator data; and

removing the first software authenticator from the first processing device responsive to receipt of the confirmation;

wherein initiating re-seeding of the second software authenticator comprises registering the second processing device with a software authenticator management server;

wherein registering the second processing device causes the software authenticator management server to generate a code which is sent to the second processing device; and

wherein the code is configured to enable re-seeding of the second software authenticator.

2. The method of claim **1**, wherein the network connection comprises one of a Bluetooth connection and a near field communication (NFC) connection.

3. The method of claim **1**, wherein the encrypting comprises:

retrieving the software authenticator data from a memory of the first processing device;

serializing the software authenticator data; and

encrypting the software authenticator data with a key derived from a user-specified password.

4. The method of claim **1**, wherein the first and second software authenticators comprise one-time passcode (OTP) generators.

5. The method of claim **1**, wherein the first and second software authenticators comprise software-implemented RSA SecurID® tokens.

6. The method of claim **1**, further comprising exchanging a binding identification between the first processing device and the second processing device, the binding identification being used to encrypt the software authenticator data.

7. The method of claim **1**, wherein the transfer of the software authenticator data from the first processing device to the second processing device does not require communication with the software authenticator management server.

8. The method of claim **1**, wherein the software authenticator data further comprises at least one of a serial number, a display interval and one or more display digits utilized by the first software authenticator provisioned on the first processing device.

9. A non-transitory processor-readable storage medium having instruction code embodied therein which when executed by a first processing device causes the first processing device:

to establish a network connection with a second processing device for transfer of software authenticator data from the first processing device to the second processing device, the software authenticator data comprising a seed value utilized by a first software authenticator provisioned on the first processing device to generate one or more passcodes;

to encrypt the software authenticator data;

to transfer the encrypted software authenticator data to the second processing device, the software authenticator data being configured to provision a second software authenticator on the second processing device;

to initiate re-seeding of the second software authenticator responsive to a successful provisioning of the second software authenticator on the second processing device;

to receive, from the second processing device, a confirmation indicating a successful transfer of the software authenticator data; and

to remove the first software authenticator from the first processing device responsive to receipt of the confirmation;

wherein initiating re-seeding of the second software authenticator comprises registering the second processing device with a software authenticator management server;

wherein registering the second processing device causes the software authenticator management server to generate a code which is sent to the second processing device; and

wherein the code is configured to enable re-seeding of the second software authenticator.

10. An apparatus comprising:

a first processing device comprising:

network interface circuitry;

a memory configured to store data associated with a first software authenticator provisioned on the first processing device; and

a processor coupled to the memory;

the first processing device under control of the processor being configured to:

establish a network connection via the network interface circuitry between the first processing device and a second processing device for transfer of software authenticator data from the first processing device to the second processing device, the software authenticator data comprising a seed value utilized by the first software authenticator provisioned on the first processing device to generate one or more passcodes;

encrypt the software authenticator data;

transfer the encrypted software authenticator data to the second processing device, the software authenticator data being configured to provision a second software authenticator on the second processing device;

initiate re-seeding of the second software authenticator responsive to a successful provisioning of the second software authenticator on the second processing device;

receive, from the second processing device, a confirmation indicating a successful transfer of the software authenticator data; and

remove the first software authenticator from the first processing device responsive to receipt of the confirmation;

wherein initiating re-seeding of the second software authenticator comprises registering the second processing device with a software authenticator management server;

wherein registering the second processing device causes the software authenticator management server to generate a code which is sent to the second processing device; and

wherein the code is configured to enable re-seeding of the second software authenticator.

11. The apparatus of claim 10, wherein the first processing device and the second processing device comprise respective source and target mobile devices.

12. The apparatus of claim 10, wherein the first processing device comprises at least one of a mobile phone, a tablet computing device and a laptop computer.

13. The apparatus of claim 10, wherein the transfer of the software authenticator data from the first processing device to the second processing device does not require communication with the software authenticator management server.

14. A method comprising:

establishing a network connection between a first processing device and a second processing device for transfer of software authenticator data from the first processing device to the second processing device, the software authenticator data comprising a seed value utilized by a first software authenticator provisioned on the first processing device to generate one or more passcodes;

receiving encrypted data from the first processing device;

decrypting the encrypted data to obtain the software authenticator data;

importing the software authenticator data into a second software authenticator stored in a memory of the second processing device;

provisioning the second software authenticator on the second processing device utilizing the software authenticator data;

re-seeding the second software authenticator responsive to a successful provisioning of the second software authenticator on the second processing device; and

sending a confirmation from the second processing device to the first processing device indicating a successful transfer of the software authenticator data;

wherein receipt of the confirmation causes removal of the first software authenticator from the first processing device;

wherein re-seeding of the second software authenticator is initiated responsive to registering the second processing device with a software authenticator management server;

wherein registering the second processing device causes the software authenticator management server to generate a code which is sent to the second processing device; and

wherein the code is configured to enable re-seeding of the second software authenticator.

15. The method of claim 14, wherein the decrypting comprises:

deriving a key from a user-specified password;

decrypting the encrypted data using the key; and

de-serializing the decrypted data.

16. The method of claim 14, further comprising exchanging a binding identification between the second processing device and the first processing device, the binding identification being used to decrypt the encrypted data.

17. The method of claim 14, wherein re-seeding the second software authenticator renders the first processing device unable to utilize the first software authenticator.

18. The method of claim 14, wherein re-seeding the second software authenticator comprises:

generating a new passcode utilizing the seed value transferred from the first processing device; and

sending the new passcode to an authentication manager, wherein the new passcode triggers a silent alarm function in the authentication manager and wherein the authentication manager associates a new seed value with the second software authenticator responsive to matching the new passcode to a derived seed value stored in the authentication manager.

19. The method of claim 14, wherein the transfer of the software authenticator data from the first processing device to the second processing device does not require communication with the software authenticator management server.

20. A non-transitory processor-readable storage medium having instruction code embodied therein which when executed by a second processing device causes the second processing device:

to establish a network connection with a first processing device for transfer of software authenticator data from the first processing device to the second processing device, the software authenticator data comprising a seed value utilized by a first software authenticator provisioned on the first processing device to generate one or more passcodes;

to receive encrypted data from the first processing device;

to decrypt the encrypted data to obtain the software authenticator data;

to import the software authenticator data into a second software authenticator stored in a memory of the second processing device;

to provision the second software authenticator on the second processing device utilizing the software authenticator data;

to re-seed the second software authenticator responsive to a successful provisioning of the second software authenticator on the second processing device; and

to send a confirmation from the second processing device to the first processing device indicating a successful transfer of the software authenticator data;

wherein receipt of the confirmation causes removal of the first software authenticator from the first processing device;

wherein re-seeding of the second software authenticator is initiated responsive to registering the second processing device with a software authenticator management server;

wherein registering the second processing device causes the software authenticator management server to generate a code which is sent to the second processing device; and

wherein the code is configured to enable re-seeding of the second software authenticator.

21. An apparatus comprising:

a first processing device comprising:

network interface circuitry;

a memory configured to store data associated with a first software authenticator; and

a processor coupled to the memory;

the first processing device under control of the processor being configured to:

establish a network connection via the network interface circuitry between the first processing device and a second processing device for transfer of data associated with a software authenticator from the second processing device to the first processing device, the software authenticator data comprising a seed value utilized by a second software authenticator provisioned on the second processing device to generate one or more passcodes;

receive encrypted data from the second processing device;

decrypt the encrypted data to obtain the software authenticator data;

import the software authenticator data into the first software authenticator stored in the memory;

provision the first software authenticator on the first processing device utilizing the software authenticator data;

re-seed the second software authenticator responsive to a successful provisioning of the second software authenticator on the second processing device; and

send a confirmation from the second processing device to the first processing device indicating a successful transfer of the software authenticator data;

wherein receipt of the confirmation causes removal of the first software authenticator from the first processing device;

wherein re-seeding of the second software authenticator is initiated responsive to registering the second processing device with a software authenticator management server;

wherein registering the second processing device causes the software authenticator management server to generate a code which is sent to the second processing device; and

wherein the code is configured to enable re-seeding of the second software authenticator.

22. The apparatus of claim **21**, wherein the first processing device and the second processing device comprise respective target and source mobile devices.

23. The apparatus of claim **21**, wherein the first processing device comprises at least one of a mobile phone, a tablet computing device and a laptop computer.

24. The apparatus of claim **21**, wherein the transfer of the software authenticator data from the first processing device to the second processing device does not require communication with the software authenticator management server.

25. The apparatus of claim **21**, wherein the first processing device is further configured to exchange a binding identification with the second processing device, the binding identification being used to encrypt the software authenticator data.

* * * * *