US 20140197234A1

(19) United States

(12) Patent Application Publication

(10) Pub. No.: US 2014/0197234 A1

(43) Pub. Date: Jul. 17, 2014

Hammad

(54) **SNAP MOBILE SECURITY APPARATUSES, METHODS AND SYSTEMS**

(71) Applicant: **VISA INTERNATIONAL SERVICE ASSOCIATION**, San Francisco, CA (US)

(72) Inventor: **Ayman Hammad**, Pleasanton, CA (US)

(73) Assignee: **VISA INTERNATIONAL SERVICE ASSOCIATION**, San Francisco, CA (US)

(21) Appl. No.: **14/216,382**

(22) Filed: **Mar. 17, 2014**

Related U.S. Application Data

(63) Continuation of application No. 13/398,817, filed on Feb. 16, 2012.

(60) Provisional application No. 61/800,012, filed on Mar. 15, 2013, provisional application No. 61/443,624, filed on Feb. 16, 2011, provisional application No. 61/512,248, filed on Jul. 27, 2011, provisional application No. 61/522,213, filed on Aug. 10, 2011, provisional application No. 61/527,576, filed on Aug. 25, 2011.

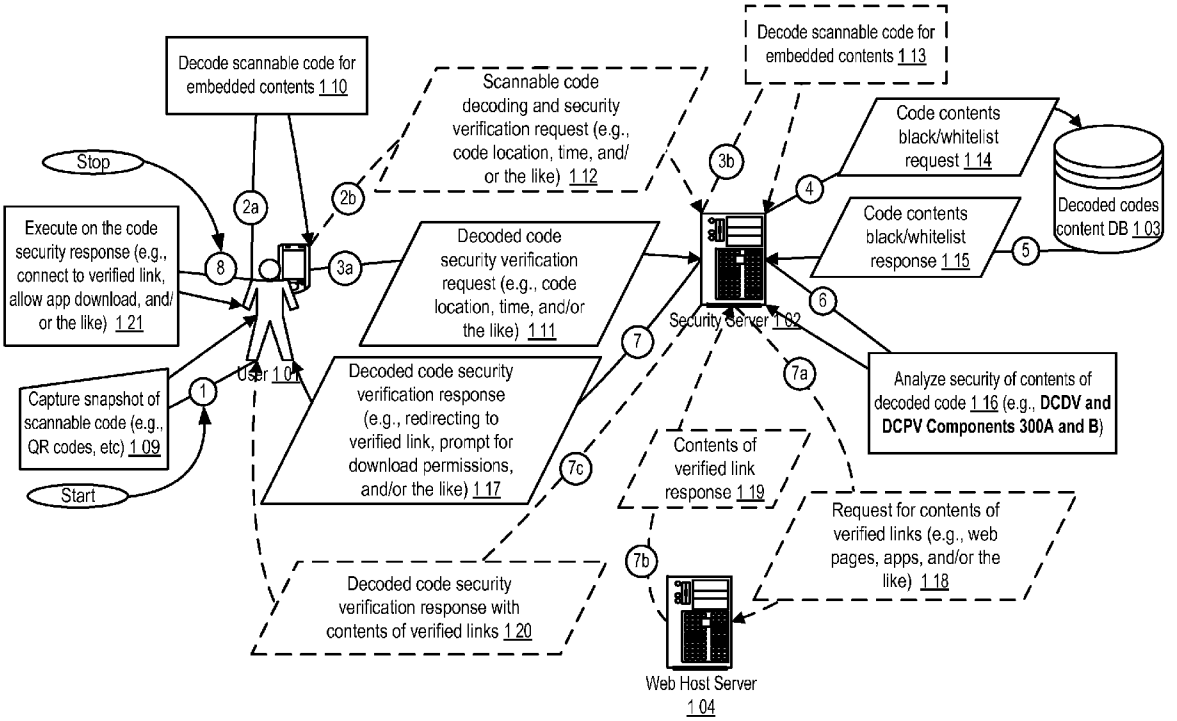
Publication Classification

(51) **Int. Cl.**
G06Q 20/36 (2006.01)

(52) **U.S. Cl.**
CPC *G06Q 20/3674* (2013.01)
USPC **235/379**

(57) **ABSTRACT**

The SNAP MOBILE SECURITY APPARATUSES, METHODS AND SYSTEMS (hereinafter "SMS") provide verification, access and security, via SMS components, to virtual wallet based electronic financial transactions. In one implementation, the SMS receives a request from a user's device to decode a scannable code and verify the security of the decoded code's contents. In some implementations, the SMS decodes the scannable code to obtain code contents requesting access to the wallet account. In some implementations, the SMS obtains from the user digital fingerprints of the user device and a request identifier for the request to access the wallet account. In some implementations, the SMS receives from the access requester digital signatures for the requester and the request identifier. In some implementations, the SMS confirms the digital fingerprints of the user device verify the device is authorized to access the wallet account. In some implementations, the SMS confirms the received digital signatures verify the access requester and the request are authorized to access the wallet account. In some implementations, the SMS sends wallet unlock and activity unlock keys to the device for activity access to the wallet.



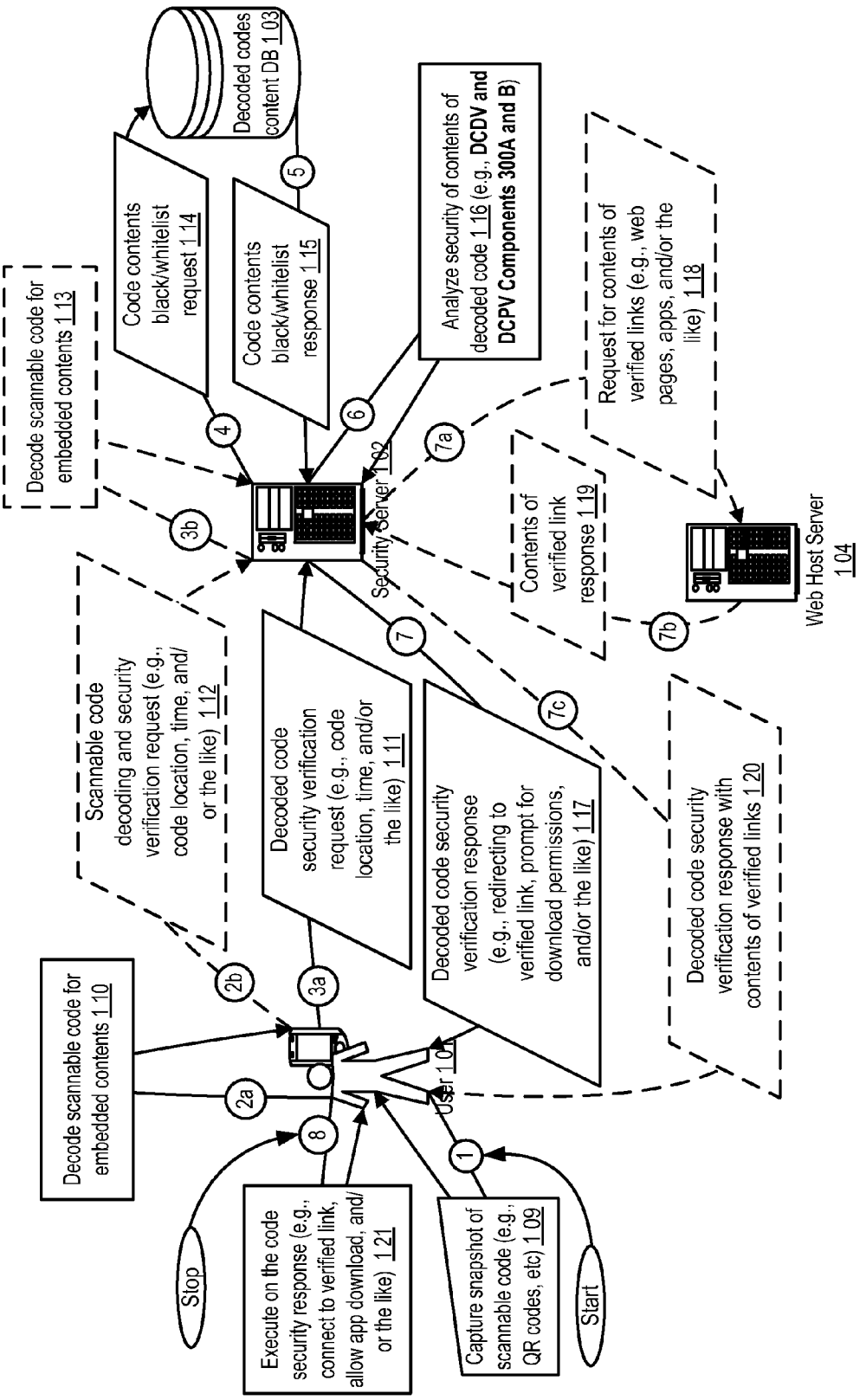

Start
Capture snapshot of scannable code (e.g., QR codes, etc) 1 09
Execute on the code security response (e.g., connect to verified link, allow app download, and/or the like) 1 21
Stop
Decode scannable code for embedded contents 1 10
User 1 01
Scannable code decoding and security verification request (e.g., code location, time, and/or the like) 1 12
Decoded code security verification request (e.g., code location, time, and/or the like) 1 11
Decoded code security verification response (e.g., redirecting to verified link, prompt for download permissions, and/or the like) 1 17
Decoded code security verification response with contents of verified links 1 20
Security Server 1 02
Web Host Server 1 04
Contents of verified link response 1 19
Request for contents of verified links (e.g., web pages, apps, and/or the like) 1 18
Analyze security of contents of decoded code 1 16 (e.g., DCDV and DCPV Components 300A and B)
Code contents black/whitelist response 1 15
Code contents black/whitelist request 1 14
Decoded codes content DB 1 03
Decode scannable code for embedded contents 1 13
1
2a
2b
3a
3b
4
5
6
7
7a
7b
7c
8

Figure 1

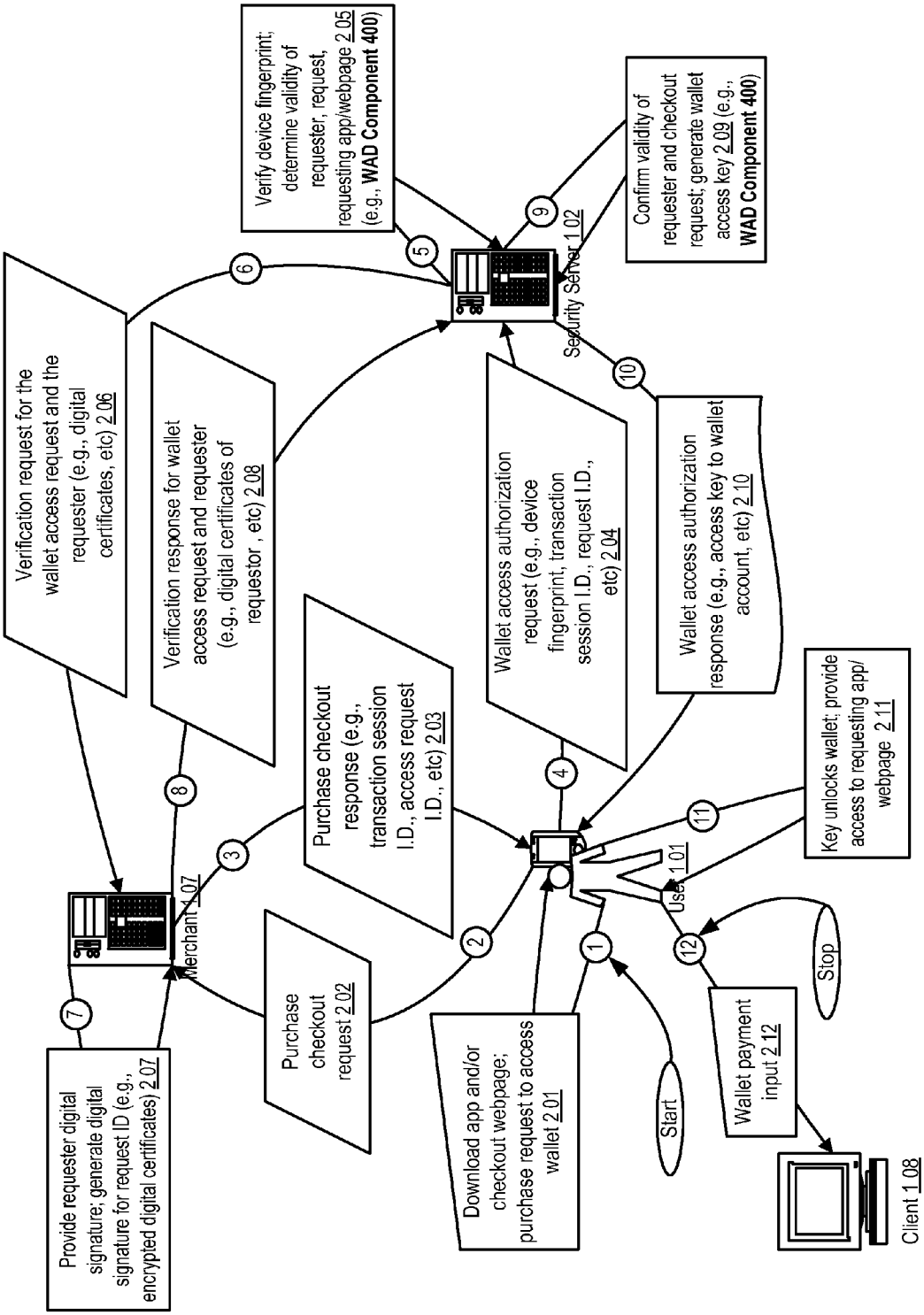

Verify device fingerprint; determine validity of requester, request, requesting app/webpage 2 05 (e.g., WAD Component 400)
Confirm validity of requester and checkout request; generate wallet access key 2 09 (e.g., WAD Component 400)
Security Server 1 02
Verification request for the wallet access request and the requester (e.g., digital certificates, etc) 2 06
Verification response for wallet access request and requester (e.g., digital certificates of requestor , etc) 2 08
Wallet access authorization request (e.g., device fingerprint, transaction session I.D., request I.D., etc) 2 04
Wallet access authorization response (e.g., access key to wallet account, etc) 2 10
Purchase checkout response (e.g., transaction session I.D., access request I.D., etc) 2 03
Merchant 1 07
Purchase checkout request 2 02
User 1 01
Key unlocks wallet; provide access to requesting app/ webpage 2 11
Provide requester digital signature; generate digital signature for request ID (e.g., encrypted digital certificates) 2 07
Download app and/or checkout webpage; purchase request to access wallet 2 01
Start
Stop
Wallet payment input 2 12
Client 1 08
1
2
3
4
5
6
7
8
9
10
11
12

Figure 2

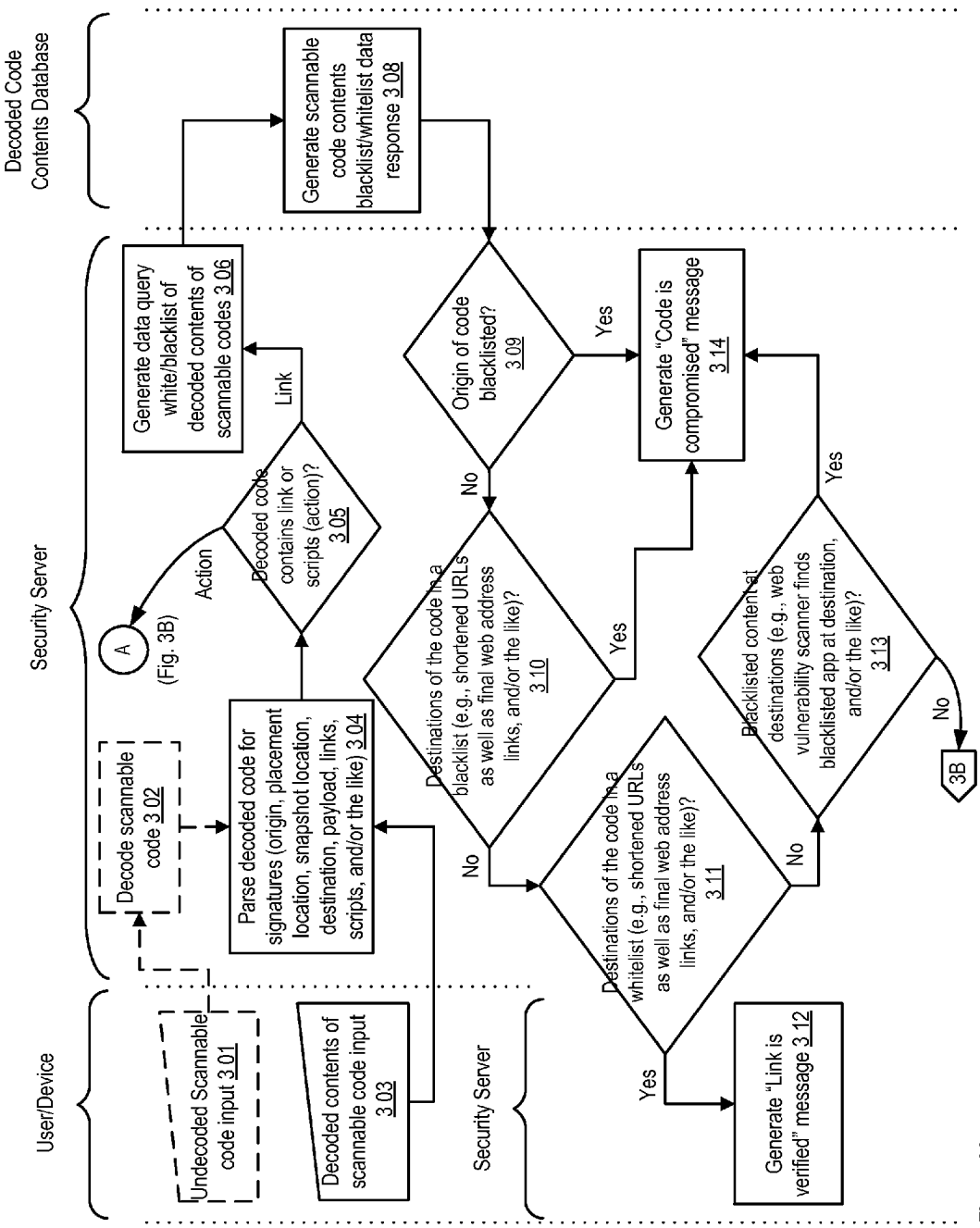

User/Device
Undecoded Scannable code input 3 01
Security Server
Decode scannable code 3 02
Parse decoded code for signatures (origin, placement location, snapshot location, destination, payload, links, scripts, and/or the like) 3 04
Decoded contents of scannable code input 3 03
Decoded code contains link or scripts (action)? 3 05
Link
Action
A
(Fig. 3B)
Generate data query white/blacklist of decoded contents of scannable codes 3 06
Decoded Code Contents Database
Generate scannable code contents blacklist/whitelist data response 3 08
Origin of code blacklisted? 3 09
Yes
No
Generate "Code is compromised" message 3 14
Destinations of the code in a blacklist (e.g., shortened URLs as well as final web address links, and/or the like)? 3 10
Yes
No
Destinations of the code in a whitelist (e.g., shortened URLs as well as final web address links, and/or the like)? 3 11
Yes
No
Generate "Link is verified" message 3 12
Blacklisted content at destinations (e.g., web vulnerability scanner finds blacklisted app at destination, and/or the like)? 3 13
Yes
No
3B

Figure 3A

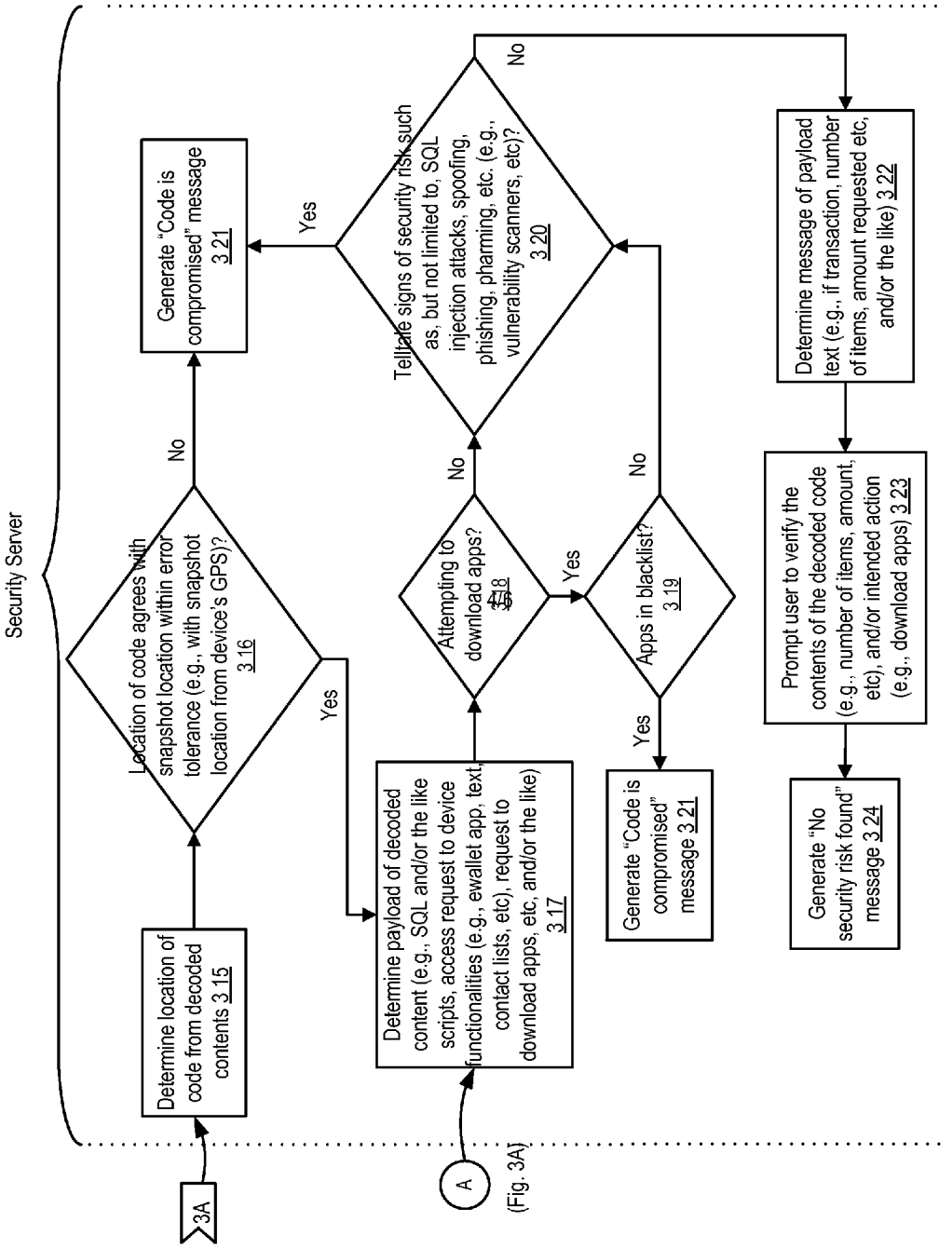

Security Server
3A
Determine location of code from decoded contents 3 15
Location of code agrees with snapshot location within error tolerance (e.g., with snapshot location from device's GPS)? 3 16
No
Yes
Generate "Code is compromised" message 3 21
Determine payload of decoded content (e.g., SQL and/or the like scripts, access request to device functionalities (e.g., ewallet app, text, contact lists, etc), request to download apps, etc, and/or the like) 3 17
A
(Fig. 3A)
Attempting to download apps? 3 18
No
Yes
Apps in blacklist? 3 19
No
Yes
Generate "Code is compromised" message 3 21
Tell-tale signs of security risk such as, but not limited to, SQL injection attacks, spoofing, phishing, pharming, etc. (e.g., vulnerability scanners, etc)? 3 20
Yes
No
Determine message of payload text (e.g., if transaction, number of items, amount requested etc, and/or the like) 3 22
Prompt user to verify the contents of the decoded code (e.g., number of items, amount, etc), and/or intended action (e.g., download apps) 3 23
Generate "No security risk found" message 3 24

Figure 3B

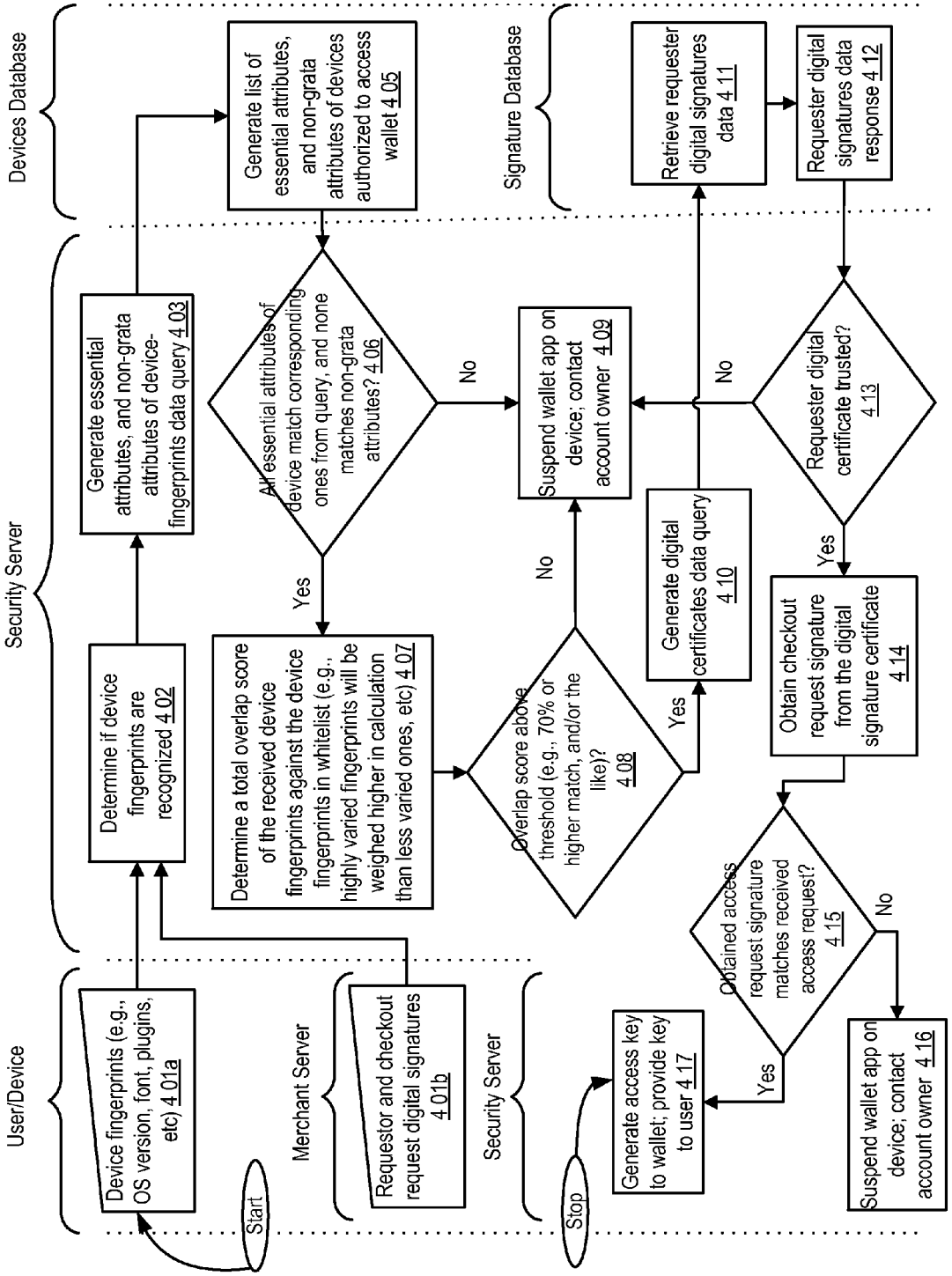

User/Device
Merchant Server
Security Server
Devices Database
Signature Database
Start
Device fingerprints (e.g., OS version, font, plugins, etc) 4 01a
Requestor and checkout request digital signatures 4 01b
Determine if device fingerprints are recognized 4 02
Generate essential attributes, and non-grata attributes of device-fingerprints data query 4 03
Generate list of essential attributes, and non-grata attributes of devices authorized to access wallet 4 05
All essential attributes of device match corresponding ones from query, and none matches non-grata attributes? 4 06
No
Yes
Determine a total overlap score of the received device fingerprints against the device fingerprints in whitelist (e.g., highly varied fingerprints will be weighed higher in calculation than less varied ones, etc) 4 07
Overlap score above threshold (e.g., 70% or higher match, and/or the like)? 4 08
No
Yes
Suspend wallet app on device; contact account owner 4 09
Generate digital certificates data query 4 10
Retrieve requester digital signatures data 4 11
Requester digital signatures data response 4 12
Requester digital certificate trusted? 4 13
No
Yes
Obtain checkout request signature from the digital signature certificate 4 14
Obtained access request signature matches received access request? 4 15
No
Yes
Suspend wallet app on device; contact account owner 4 16
Generate access key to wallet; provide key to user 4 17
Stop

Figure 4

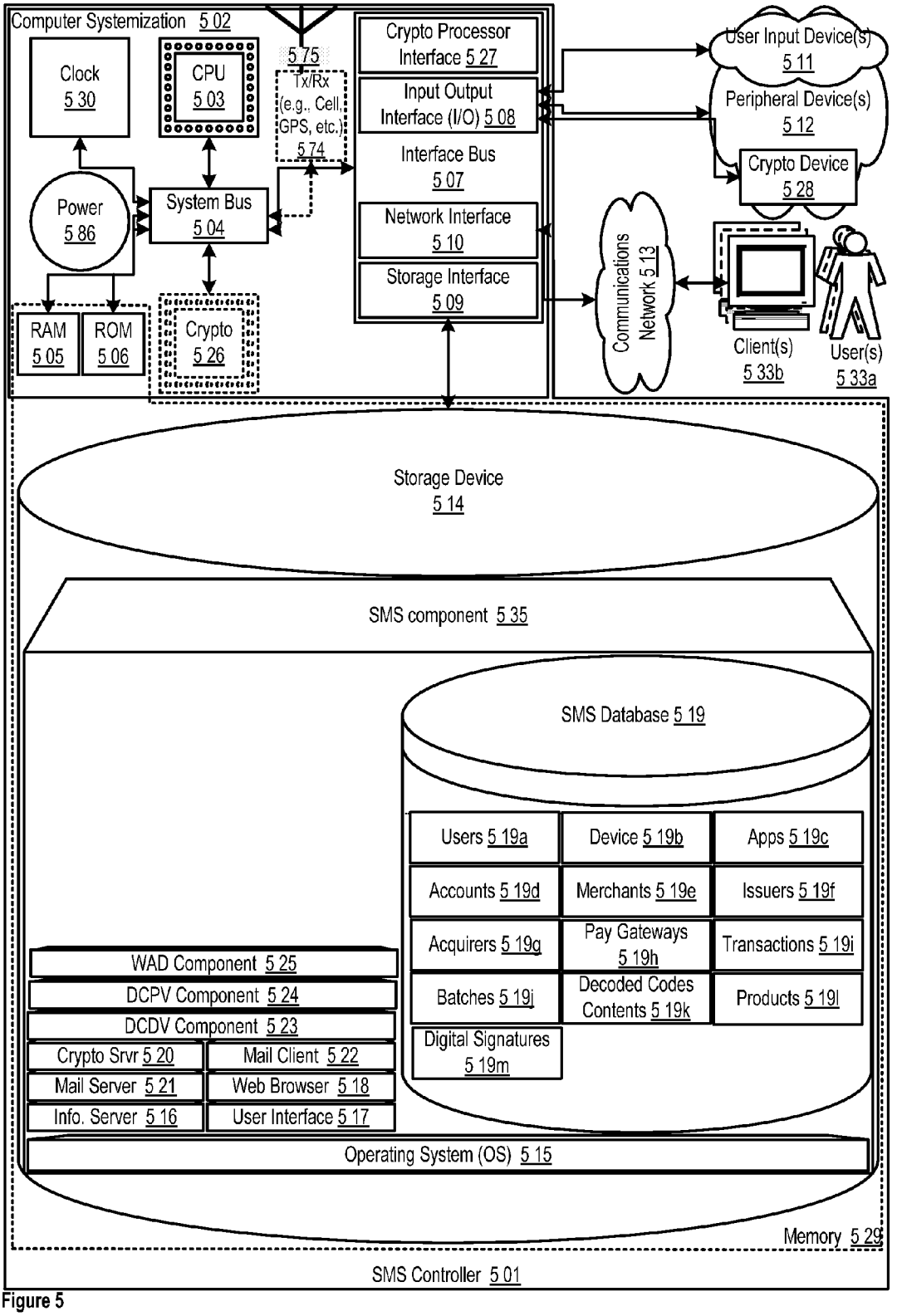

Computer Systemization 5 02
Clock 5 30
CPU 5 03
5 75
Tx/Rx (e.g., Cell, GPS, etc.) 5 74
Crypto Processor Interface 5 27
Input Output Interface (I/O) 5 08
Interface Bus 5 07
Network Interface 5 10
Storage Interface 5 09
User Input Device(s) 5 11
Peripheral Device(s) 5 12
Crypto Device 5 28
Power 5 86
System Bus 5 04
RAM 5 05
ROM 5 06
Crypto 5 26
Communications Network 5 13
Client(s) 5 33b
User(s) 5 33a
Storage Device 5 14
SMS component 5 35
SMS Database 5 19
Users 5 19a
Device 5 19b
Apps 5 19c
Accounts 5 19d
Merchants 5 19e
Issuers 5 19f
Acquirers 5 19g
Pay Gateways 5 19h
Transactions 5 19i
Batches 5 19j
Decoded Codes Contents 5 19k
Products 5 19l
Digital Signatures 5 19m
WAD Component 5 25
DCPV Component 5 24
DCDV Component 5 23
Crypto Srvr 5 20
Mail Client 5 22
Mail Server 5 21
Web Browser 5 18
Info. Server 5 16
User Interface 5 17
Operating System (OS) 5 15
Memory 5 29
SMS Controller 5 01

Figure 5

SNAP MOBILE SECURITY APPARATUSES, METHODS AND SYSTEMS

[0001] This patent for letters patent document discloses and describes various novel innovations and inventive aspects of SNAP MOBILE SECURITY technology (hereinafter "disclosure") and contains material that is subject to copyright, mask work, and/or other intellectual property protection. The respective owners of such intellectual property have no objection to the facsimile reproduction of the disclosure by anyone as it appears in published Patent Office file/records, but otherwise reserve all rights.

PRIORITY CLAIM

[0002] This application claims priority to U.S. patent application Ser. No. 61/800,012, filed Mar. 15, 2013 and entitled "Snap Mobile Security Apparatuses, Methods and Systems," attorney docket 349US01. This application also claims priority to: U.S. application Ser. No. 13/398,817 filed Feb. 16, 2012, entitled "SNAP MOBILE PAYMENT APPARATUSES, METHODS AND SYSTEMS," attorney docket P-42032US01, which claims priority to U.S. provisional patent application Ser. No. 61/443,624 filed Feb. 16, 2011, entitled "MOBILE CAPTURE CHECKOUT APPARATUSES, METHODS AND SYSTEMS," attorney docket no. P-42032PRV; U.S. provisional patent application Ser. No. 61/512,248 filed Jul. 27, 2011, entitled "SNAP MOBILE PAYMENT APPARATUSES, METHODS AND SYSTEMS," attorney docket no. 10US01; U.S. provisional patent application Ser. No. 61/522,213 filed Aug. 10, 2011, entitled "UNIVERSAL MOBILE PAYMENT PLATFORM APPARATUSES, METHODS AND SYSTEMS," attorney docket no. 10US03; and U.S. provisional patent application Ser. No. 61/527,576 filed Aug. 25, 2011, entitled "SNAP MOBILE PAYMENT APPARATUSES, METHODS AND SYSTEMS," attorney docket no. 10US02. The entire contents of the aforementioned applications are expressly incorporated by reference herein.

FIELD

[0003] The present innovations generally address apparatuses, methods, and systems for electronic purchase transactions, and more particularly, include SNAP MOBILE SECURITY APPARATUSES, METHODS AND SYSTEMS ("SMS").

BACKGROUND

[0004] Consumer transactions require a customer to select a product from a store shelf or a website, and then to check them out at a checkout counter or a webpage. Product information may be entered automatically by scanning an item barcode with an integrated barcode scanner, and the customer is usually provided with a number of payment options, such as cash, check, credit card or debit card to pay for the purchase.

SUMMARY

[0005] In accordance with the teachings provided herein, systems, methods, non-transitory computer-readable medium, and apparatuses are disclosed for operation upon data processing devices for providing mobile security, such as to: receive, through one or more processors, a request from a user's device to decode a scannable code and verify the security of the decoded code's contents; decode, through the one or more processors, the scannable code to obtain code contents requesting access to the wallet account; obtain from the user, through the one or more processors, digital fingerprints of the user device and a request identifier for the request to access the wallet account; receive from the access requester, through the one or more processors, digital signatures for the requester and the request identifier; confirm, through the one or more processors, that the digital fingerprints of the user device verify the device is authorized to access the wallet account; confirm, through the one or more processors, the received digital signatures verify the access requester and the request are authorized to access the wallet account; and send, through the one or more processors, wallet unlock and activity unlock keys to the device for activity access to the wallet.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] The accompanying appendices, drawings, figures, images, etc. illustrate various example, non-limiting, inventive aspects, embodiments, and features ("e.g.," or "example (s)") in accordance with the present disclosure:

[0007] FIG. **1** shows a datagraph diagram illustrating example features of the SMS verifying the security contents of decoded scannable codes;

[0008] FIG. **2** shows a datagraph diagram illustrating example features of the SMS validating authorization requests for access to a wallet account;

[0009] FIGS. **3**A-B show logic flow diagrams illustrating example features of the SMS verifying the security of contents of decoded scannable codes;

[0010] FIG. **4** shows a logic flow diagram illustrating example features of the SMS validating authorization requests for access to a wallet account; and

[0011] FIG. **5** shows a block diagram illustrating examples of a SMS controller.

[0012] The leading number of each reference number within the drawings indicates the figure in which that reference number is introduced and/or detailed. As such, a detailed discussion of reference number **101** would be found and/or introduced in FIG. **1**. Reference number **201** is introduced in FIG. **2**, etc.

DETAILED DESCRIPTION

Snap Mobile Security (SMS)

[0013] The SNAP MOBILE SECURITY APPARATUSES, METHODS AND SYSTEMS (hereinafter "SMS") provide verification, access and security, via SMS components, to virtual wallet based electronic financial transactions.

[0014] FIG. **1** shows a datagraph diagram illustrating example features of the SMS verifying the security contents of decoded scannable codes. In some implementations, a user **101** may take a snapshot of a scannable code such as, but not limited to, a quick response (QR) code, e.g., **109**. For example, the user may utilize a device such as a smartphone to capture an image of the code. In some implementations, the user may decode the code at the user's computing device, e.g., **110**. In some implementations, the user may send the contents of the decoded code to a security server **102** to verify the validity and security of the contents. In some implementations, the user may decide to transfer the undecoded snapshot to the server for decoding, and the security server may decode the code, e.g., **113**. In such implementations, the user may

request in the security server to decode the undecoded code and verify the validity and security of the contents of the decoded code, e.g., **112**. In some implementations, some scannable codes, though advertised as facilitators of legitimate transactions, may expose users to security risks and fraud such as phishing, pharming, and/or the like. For example, the contents of a nefarious decoded code may contain a Uniform Resource Locator (URL) link that leads to fraudulent websites that may expose a user to unwanted/unsolicited content (e.g., ads, etc), trick a user into providing sensitive personal information, attempt to download unwanted/unsolicited material (e.g., malicious apps, etc) onto a user's device, and/or the like. For example, the decoded code may compromise the security of a user's device by subjecting it to malicious attacks such as SQL injections, and/or the like.

[0015] In some implementations, a user's device and/or a security server may decode the snapshot of a scannable code, such as, but not limited to, a QR code. An example listing of a verification request **111**, substantially in the form of a HTTP(S) POST message including XML-formatted data, is provided below:

```
POST /verificationrequest.php HTTP/1.1
Host: www.security.com
Content-Type: Application/XML
Content-Length: 667
<?XML version = "1.0" encoding = "UTF-8"?>
<qrverify_request>
    <timestamp>2011-04-01 :23:59:59</timestamp>
    <transaction amount>$660.89</transaction amount>
    <digital_sign>
            45e2085fa20496c91df574dc5652e145
    </digital_sign>
    <QRCodePayload>
            <location_link>www.phishpharm.com</location_link>
            <merchant_id>AE783</merchant_id>
            <merchant_name>Scammer, Inc. </merchant_ name>
            <store_id>88234</store_id>
            <post_location>6th Ave and 42nd St</post_location>
            <transaction_id>AFE 1213344</transaction_id>
    </QRCodePayload>
    //<QRCodePayload>
    //          <image_data>JPEGDATA</imagedata>
    //</QRCodePayload>
</qrverify_request>
```

[0016] In some implementations, the SMS may determine if the contents of the decoded code pose any security risk to the user device. For example, the SMS may compare the decoded code contents to black and white lists of code contents, and determine if the decoded contents pose some or no security risk to user's device, respectively. For example, the security server may issue PHP/SQL commands to query a database table (such as FIG. **5**, Decoded Codes Contents database **519***k*) for blacklist/whitelist code contents data. An example code contents blacklist/whitelist query **114**, substantially in the form of PHP/SQL commands, is provided below:

```
<?PHP
header('Content-Type: text/plain');
mysql_connect("254.93.179.112",$DBserver,$password); // access
database server
mysql_select_db("SMS_DB.SQL"); // select database table to search
//create query
```

-continued

```
$query = "SELECT blacklist whitelist FROM CodeContentsTable
WHERE QRlists LIKE
    '%' $QRCodePayload";
$result = mysql_query($query); // perform the search query
mysql_close("SMS_DB.SQL"); // close database access
?>
```

[0017] In some implementations, once receiving the blacklist/whitelist of scannable codes, the SMS may initiate the steps to verify the security of the decoded code, e.g., **116**. For example, the server may merely redirect the user to the link for the user to execute the link, e.g. **117**. For example, the security server may provide a redirected link response to user device as a HTTP(S) POST message including XML-formatted data. An example listing of a redirected link response **117**, substantially in the form of a HTTP(S) POST message including XML-formatted data, is provided below:

```
POST /verificationresponse.php HTTP/1.1
Host: www.userdevice.com
Content-Type: Application/XML
Content-Length: 667
<?XML version = "1.0" encoding = "UTF-8"?>
<qrverify_response>
    <timestamp>2011-04-01 :23:59:59</timestamp>
    <QRCodePayload>
        <redirected_link>www.verifiedlink.com</redirected_link>
        <merchant_id>AE783</merchant_id>
        <merchant_name>Legit Business, Inc. </merchant_ name>
        <store_id>88234</store_id>
        <transaction_id>AFE 1213344</transaction_id>
    </QRCodePayload>
</qrverify_response>
```

[0018] For example, the user may decide to launch a website, and/or download an app. In some embodiments, the SMS may retrieve links found in the decoded code and determine, via SMS components, the links are secure and that the user may launch the URL link. For example, the server may fetch the destination of the link, e.g. **118**, and provide the link destination to the user, e.g., **119**. For example, the server may launch the URL link and open a webpage on the user's device. For example, the security server may provide a webpage request to a web hosting server **104** as a HTTP(S) GET message including XML-formatted data. An example listing of a webpage request **118**, substantially in the form of a HTTP(S) GET message including XML-formatted data, is provided below:

```
GET /page.php/ HTTP/1.1
Host: www.site.com
User-Agent: Mozilla/5.0
Accept: text/html,application/xhtml+xml,application/xml;
Accept-Language: en-us,en;
Accept-Charset: ISO-8859-1,utf-8;
Cookie: PHPSESSID=r2t5uvjq435r4q7ib3vtdjq120
```

[0019] In some embodiments, the web hosting server may respond back to the security server with the requested destination of the link. For example, the web hosting server may provide a webpage in response to the security server as a HTTP(S) POST message including XML-formatted data. The webpage contents may then be relayed to the user **120**.

[0020] In some embodiments, the SMS may initiate the steps to verify that the user of the device is in fact an authorized user, and that the device is secure, i.e., its security is not compromised, e.g., **121**. Upon confirming that the user is authorized, and the device is secure, e.g., **130**, in such embodiments, the user may execute on the response received from the security server, e.g., **131**.

[0021] FIG. **2** shows a datagraph diagram illustrating example features of the SMS validating authorization requests for access to a wallet account. In some embodiments, a user may take a snapshot of a scannable code such as, but not limited to a QR code, and have the user's device and/or the security server decode it. Upon verifying the security of the decoded contents of the code, in some implementations, the user may execute on the decoded contents. For example, the codes may be item codes for products, and the user may launch a URL link, and/or download and launch an app to initiate a purchase request. In some embodiments, the user may wish to provide a checkout request to the merchant server **107**. For example, the checkout request to the merchant server may be a HTTP(S) POST message including XML-formatted data. An example listing of a checkout request **202**, substantially in the form of a HTTP(S) POST message including XML-formatted data, is provided below:

```
POST /checkoutrequest.php HTTP/1.1
Host: www.merchant.com
Content-Type: Application/XML
Content-Length: 667
<?XML version = "1.0" encoding = "UTF-8"?>
<QR_data>
    <order_ID>4NFU4RG94</order_ID>
    <timestamp>2035-02-22 15:22:43</timestamp>
    <expiry_lapse>00:01:00</expiry_lapse>
    <total_cost>$74.46</total_cost>
    <user_id>john.q@gmail.com</user_id>
    <secure_element>www.merchant.com/securedyn/xyz/123.png</secure_element>
    <merchant_params>
        <merchant_id>54TBRELF8</merchant_id>
        <merchant_name>BIG_APPLE_BOOKSTORE</merchant_name>
        <address> 1 Piazza Square </address>
        <city> New York </city>
        <zip_code> 10001 </zip_code>
        <merchant_auth_key>TMN45GER98</merchant_auth_key>
    </merchant_params>
    <purchase_detail>
        <cart>
            <product>
                <product_type>book</product_type>
                <product_params>
                    <product_title>Blood Meridian</product_title>
                    <ISBN>0-394-54482-X</ISBN>
                    <edition>1st ed.</edition>
                    <cover>hardbound</cover>
                </product_params>
                <quantity>1</quantity>
                <unit_cost>$74.46</unit_cost>
            </product>
        </cart>
    </purchase_detail>
</QR_data>
```

[0022] In response, in some embodiments, the merchant server may provide the user with data such as, but not limited to, the transaction session I.D., access request I.D., requestor I.D., and/or the like, e.g., **203**. For example, the checkout response to the merchant server may be a HTTP(S) POST message including XML-formatted data. An example listing of a checkout response **203**, substantially in the form of a HTTP(S) POST message including XML-formatted data, is provided below:

```
POST /checkoutresponse.php HTTP/1.1
Host: www.userdevice.com
Content-Type: Application/XML
Content-Length: 667
<?XML version = "1.0" encoding = "UTF-8"?>
<checkout_response>
    <session_ID>4NFU4RG94</session_ID>
    <timestamp>2035-02-22 15:22:43</timestamp>
    <total_cost>$74.46</total_cost>
    <user_id>john.q@gmail.com</user_id>
    <access_auth>
        <access_request_ID>
            <timestamp>2035-02-22 15:25:43</timestamp>
            <amount>$74.46</amount>
            <merchant_id>54TBRELF8</merchant_id>
            <session_ID>4NFU4RG94</session_ID>
            <consumer_acct_chrg_access> VISA ************5634
        </consumer_acct_chrg_access>
        </access_request_ID>
        <access_requester_ID>Big Firm, Inc</ access_requester_ID>
    <purchase_detail>
        <cart>
```

-continued

```
            <product>
                <product_type>book</product_type>
                <product_params>
                    <product_title>Blood Meridian</product_title>
                    <ISBN>0-394-54482-X</ISBN>
```

-continued

```
<edition>1st ed.</edition>
<cover>hardbound</cover>
```

like), request identifier, requester identifier, and/or the like. For example, the access authorization request to the security server may be a HTTP(S) POST message including XML-formatted data. An example listing of an access authorization request **204**, substantially in the form of a HTTP(S) POST message including XML-formatted data, is provided below:

```
POST /accessauthorization.php HTTP/1.1
Host: www.securityserver.com
Content-Type: Application/XML
Content-Length: 667
<?XML version = "1.0" encoding = "UTF-8"?>
<access_auth>
    <access_request_ID>
        <timestamp>2035-02-22 15:25:43</timestamp>
        <amount>$74.46</amount>
        <merchant_id>54TBRELF8</merchant_id>
        <session_ID>4NFU4RG94</session_ID>
        <consumer_acct_chrg_access> VISA ************5634
        </consumer_acct_chrg_access>
    </access_request_ID>
    <access_requester_ID>Big Firm, Inc</ access_requester_ID>
    <timestamp>2035-02-22 15:22:43</timestamp>
    <secure_element>www.merchant.com/securedyn/xyz/123.png</secure_element>
    <device_fingerprints>
        <OS> Windows </OS>
        <user_agent> Mozilla </user_agent>
        <http_accept_info>
            <info_1> text/html </info_1>
            <info_2> application/xhtml+xml </info_2>
            ...
        </ http_accept_info >
        <plug_ins>
            <Flash_Version> 11.1.102.55 </Flash_Version>
            <Adobe_Reader> 10.1.2.45 </Adobe_Reader>
            ...
        </plug_ins>
        <fonts> ... </fonts>
        <screen_dim> 1920X1200X24 </screen_dim>
    </ device_fingerprints>
</ access_auth >
```

-continued

```
                </product_params>
                <quantity>1</quantity>
                <unit_cost>$74.46</unit_cost>
            </product>
        </cart>
    </purchase_detail>
</checkout_response >
```

[0023] In some implementations, the webpage at the URL link and/or the launched app may request access to the wallet account on the device to initiate payment for the purchase, e.g., **201**. In some implementations, the SMS may initiate a verification process to confirm that only authorized entities have access to the wallet app. For example, the SMS may verify the access requestor is authorized to access the wallet app. In some implementations, the SMS may verify the validity of the checkout request, and the app/webpage making the request. In some embodiments, the SMS may determine that the user device from which the access request is coming from is an authorized device. In some implementations, the SMS may forward the received data, along with a wallet access authorization request, to the security server to verify the requester, the access request, and the security of the user device. For example, the authorization request may include data such as fingerprints of user's device (e.g., user agent (operating systems, browsers, toolbars, etc), fonts, plugin versions, screen size and resolution, time zone, and/or the

[0024] Upon receiving the access authorization request, in some implementations, the security server may verify the user device is authorized to access the wallet app, e.g., **205**. For example, the server may calculate a total weighed overlap score between the received device fingerprints and those that are whitelisted as safe. For example, those attributes that have a large variety, (e.g., fonts, etc) may be weighed much higher those with less variety (e.g., operating system, etc). The higher the score is the more it indicates the user device belongs in the whitelist, and may be verified as a device authorized to access the wallet app. In some implementations, once the user device is established as an authorized device, the security server may initiate a request to the merchant server to verify the access requester is authorized to access the app and the request is a legitimate one, e.g., **206**. For example, the security server may provide a verification request for request I.D. and requester to the merchant server as a HTTP (S) POST message including XML-formatted data. An example listing of a verification request for request I.D. and requester **206**, substantially in the form of a HTTP(S) POST message including XML-formatted data, is provided below:

```
POST /verifyaccess.php HTTP/1.1
Host: www.merchant.com
Content-Type: Application/XML
Content-Length: 667
```

-continued

```
<?XML version = "1.0" encoding = "UTF-8"?>
<access_verify>
    <access_request_ID>
        <timestamp>2035-02-22 15:25:43</timestamp>
        <amount>$74.46</amount>
        <merchant_id>54TBRELF8</merchant_id>
        <session_ID>4NFU4RG94</session_ID>
        <consumer_acct_chrg_access> VISA **********5634
        </consumer_acct_chrg_access>
    </access_request_ID>
    <access_requester_ID>Big Firm, Inc</ access_requester_ID>
    <timestamp>2035-02-22 15:22:43</timestamp>
</access_verify>
```

[0025] For example, the security server may query for the digital signature of the requester, and a digital signature for the request identifier. Upon generating a digital signature for the request identifier, e.g., **207**, the merchant server may verification response to the security server as a HTTP(S) POST message including XML-formatted data. An example listing of a verification request for request I.D. and requester **208**, substantially in the form of a HTTP(S) POST message including XML-formatted data, is provided below:

```
POST /digicert.php HTTP/1.1
Host: www.security.com
Content-Type: Application/XML
Content-Length: 667
<?XML version = "1.0" encoding = "UTF-8"?>
<access_verify>
    <timestamp>2035-02-22 15:22:43</timestamp>
    <digicert_requester>
        // DigiCert file for requester's digital certificate
        DigiCert:: cert($data,'requester.cert');
    </digicert_requester>
    <digicert_request>
        // DigiCert file for for request digital signature
        DigiCert::cert($data, 'requestid.cert);
    </digicert_request>
</access_verify>
```

[0026] Upon receiving the digital certificates, in some embodiments, the security server may determine if the request is legitimate, and the requester is authorized to access the wallet app, e.g., **209**. For example, with the latter, the server may compare the requester's digital signature with ones in a whitelist, and determine if the requester is approved. In some implementations, the server may retrieve the digital signature of the request and compare the retrieved request identifier with the one received from the user's device. In some implementations, once the requester and the user device are verified as entities authorized to access the wallet app, and the access request is confirmed as legitimate, the security server may generate a wallet access key to supply **210** to the user's device to unlock the wallet app, and allow the request access to the wallet app. For example, the security server may provide a wallet access authorization response to the user device as a HTTP(S) POST message including XML-formatted data. An example listing of a wallet access authorization response **210**, substantially in the form of a HTTP(S) POST message including XML-formatted data, is provided below:

```
POST /accessauthorization.php HTTP/1.1
Host: www.merchant.com
Content-Type: Application/XML
```

-continued

```
Content-Length: 667
<?XML version = "1.0" encoding = "UTF-8"?>
<access_auth>
    <wallet_access> TRUE </wallet_access>
    <timestamp>2035-02-22 15:25:43</timestamp>
    <authorization_id>KJ789BJK90743GJH</authorization_id>
    <session_ID>4NFU4RG94</session_ID>
    <wallet_key>54TBRELF8</wallet_key>
    <action_key>4NFU4RG94</action_key>
</access_auth>
```

For example, the webpage and/or the app that requested access to the wallet app may launch the wallet app to initiate the payment process, e.g., **212**.

[0027] FIGS. **3**A-B show logic flow diagrams illustrating example features of the SMS verifying the security of contents of decoded scannable codes. With reference to FIG. **3**A, in some embodiments, a user's device may capture a snapshot of a scannable code such as, but not limited to, a QR code, and send, e.g., **301**, the undecoded code to a security server for decoding, e.g., **302**. In some embodiments, the user's device may decode the snapshot, e.g., **303**, and send the decoded contents to the security server. In some implementations, the security server may parse through the decoded code contents, and collect the signatures of the code such as, but not limited to, the origin of the code, the placement (e.g., public street, merchant location, etc), links in the code, number of items, amounts, and/or the like, e.g., **304**. In some implementations, the server may query the decoded code contents database (such as FIG. **5**, Decoded Code Contents **519***k*) for lists of decoded code contents that belong to a whitelist, and to a blacklist, e.g., **306** to determine the whitelist/blacklist status of the contents of the decoded code, e.g., **305**. For example, the server may check if the origin of the code is blacklisted **309**, any of the destinations (e.g., links, etc) are blacklisted **310** and/or whitelisted **311**, etc. In some embodiments, one, some or all decoded contents may be blacklisted, and the server may generate a message announcing to the device user that the code is compromised and not to be trusted, e.g., **314**. In some embodiments, the destinations may not be in a blacklist, but there may be contents at the link that are blacklisted, e.g., **313**. For example, the link may contain a blacklisted app. In these embodiments, the server may generate a message announcing to the device user that the code is compromised and not to be trusted, e.g., **314**.

[0028] With reference to FIG. **3**B, in some implementations, the security server may determine the location of the code from parsing through the decoded contents, e.g., **315**, and compare that to the location at which the snapped code was found at, e.g., **316**. For example, the location of the decoded code as gleaned from the decoded contents may be compared to the GPS position of the device when the snapshot was taken. In some implementations, the two locations may not match, suggesting that the decoded code should not have been at the location and may be fraudulent. For example, a fraudulent QR code may have been placed over a legitimate one, in a "attack-in-the-middle" scenario. In these implementations, the server may contact the device user with a message that the code is compromised, e.g., **321**. In some implementations, the decoded code may attempt to download an app, e.g., **318**, and/or request access to the functionalities of the device (e.g., contact lists, email access, texting, apps, etc), e.g., **317**. In some implementations, the server may discover the activities of the decoded code may signal security com-

promise. For example, a vulnerability scanning session may discover signs of attacks such as command injections, etc, scams such as phishing, pharming, etc, e.g., **320**. In these embodiments, the security server may contact the device user to warn that the snapped code's security is compromised, e.g., **321**.

[0029] FIG. **4** shows a logic flow diagram illustrating example features of the SMS validating authorization requests for access to a wallet account. In some implementations, the security server may obtain from the user's device data on the device's fingerprints, e.g., **401***a*. For example, the data may include user agent (operating systems, browsers, toolbars, etc), fonts, plugin versions, screen dimensions and resolution, time zones, and/or the like. In some embodiments, the security server may receive from the merchant server a digital signature of the wallet access requester. In some embodiments, the merchant server may generate an encrypted digital signature certificate for the access request identifier, and pass along the digital signature to the security server, e.g., **401***b*. Upon obtaining the device fingerprints, in some implementations, the security server may generate a query from a database table (such as FIG. **5**, Devices **519***b*) for a list of essential attributes every authorized device should have, and likewise non-grata attributes any of which will result in a device being blocked from accessing the wallet account. In some implementations, the server may ascertain all the essential attributes of the received device fingerprints match the corresponding essential attributes from the query, and no attribute matches the non-grata attributes, e.g., **406**. For example, if only mobile devices are authorized to access the wallet account (i.e. large screens are non-grata), the security server may ascertain that the received device fingerprints show, to some predetermined confidence level, that the device is a mobile (i.e., small screen) device. In these embodiments, if the device is found to not satisfy this condition, despite matching all the other attributes in the whitelist, the SMS may suspend the wallet app on the device and contact the account owner to communicate the security risk, e.g., **409**. In some implementations, the device may satisfy the condition, and the server may resort to calculating the overall commonalities of the received device fingerprints and the fingerprints identified in the whitelist, e.g., **407**. For example, if the commonalities (i.e. overlap) exceed some threshold, the server may recognize the device as authorized to access the wallet account.

[0030] With the user's device recognized as an authorized device to access the wallet account, in some implementations, the security server may determine if the received digital signatures for the access requester and the request identifier are legitimate. In some embodiments, the server may generate a query to a database table (such as FIG. **5**, Digital Signatures **519***m*) for the whitelist of digital signatures of the access requesters, e.g., **410**. For example, if the received digital signature of the requester matches any in the whitelist, the server may decide the requester is legitimate, e.g., **413**. With the verification of the requester accomplished, in some implementations, the security server may retrieve **414** the access request identifier from the digital signature certificate, and compare the identifier to the one received from the user device, e.g., **415**. If there is a match, the server may generate a wallet account access key to grant the requester access to the wallet app on the user's device, e.g., **417**.

SMS Controller

[0031] FIG. **5** shows a block diagram illustrating examples of a SMS controller **501**. In this embodiment, the SMS controller **501** may serve to aggregate, process, store, search, serve, identify, instruct, generate, match, and/or facilitate interactions with a computer through various technologies, and/or other related data.

[0032] Users, e.g., **533***a*, which may be people and/or other systems, may engage information technology systems (e.g., computers) to facilitate information processing. In turn, computers employ processors to process information; such processors **503** may be referred to as central processing units (CPU). One form of processor is referred to as a microprocessor. CPUs use communicative circuits to pass binary encoded signals acting as instructions to enable various operations. These instructions may be operational and/or data instructions containing and/or referencing other instructions and data in various processor accessible and operable areas of memory **529** (e.g., registers, cache memory, random access memory, etc.). Such communicative instructions may be stored and/or transmitted in batches (e.g., batches of instructions) as programs and/or data components to facilitate desired operations. These stored instruction codes, e.g., programs, may engage the CPU circuit components and other motherboard and/or system components to perform desired operations. One type of program is a computer operating system, which, may be executed by CPU on a computer; the operating system enables and facilitates users to access and operate computer information technology and resources. Some resources that may be employed in information technology systems include: input and output mechanisms through which data may pass into and out of a computer; memory storage into which data may be saved; and processors by which information may be processed. These information technology systems may be used to collect data for later retrieval, analysis, and manipulation, which may be facilitated through a database program. These information technology systems provide interfaces that allow users to access and operate various system components.

[0033] In one embodiment, the SMS controller **501** may be connected to and/or communicate with entities such as, but not limited to: one or more users from user input devices **511**; peripheral devices **512**; an optional cryptographic processor device **528**; and/or a communications network **513**. For example, the SMS controller **501** may be connected to and/or communicate with users, e.g., **533***a*, operating client device (s), e.g., **533***b*, including, but not limited to, personal computer(s), server(s) and/or various mobile device(s) including, but not limited to, cellular telephone(s), smartphone(s) (e.g., iPhone®, Blackberry®, Android OS-based phones etc.), tablet computer(s) (e.g., Apple iPad™, HP Slate™, Motorola Xoom™, etc.), eBook reader(s) (e.g., Amazon Kindle™, Barnes and Noble's Nook™ eReader, etc.), laptop computer (s), notebook(s), netbook(s), gaming console(s) (e.g., XBOX Live™, Nintendo® DS, Sony PlayStation® Portable, etc.), portable scanner(s), and/or the like.

[0034] Networks are commonly thought to comprise the interconnection and interoperation of clients, servers, and intermediary nodes in a graph topology. It should be noted that the term "server" as used throughout this application refers generally to a computer, other device, program, or combination thereof that processes and responds to the requests of remote users across a communications network. Servers serve their information to requesting "clients." The

term "client" as used herein refers generally to a computer, program, other device, user and/or combination thereof that is capable of processing and making requests and obtaining and processing any responses from servers across a communications network. A computer, other device, program, or combination thereof that facilitates, processes information and requests, and/or furthers the passage of information from a source user to a destination user is commonly referred to as a "node." Networks are generally thought to facilitate the transfer of information from source points to destinations. A node specifically tasked with furthering the passage of information from a source to a destination is commonly called a "router." There are many forms of networks such as Local Area Networks (LANs), Pico networks, Wide Area Networks (WANs), Wireless Networks (WLANs), etc. For example, the Internet is generally accepted as being an interconnection of a multitude of networks whereby remote clients and servers may access and interoperate with one another.

[0035] The SMS controller **501** may be based on computer systems that may comprise, but are not limited to, components such as: a computer systemization **502** connected to memory **529**.

Computer Systemization

[0036] A computer systemization **502** may comprise a clock **530**, central processing unit ("CPU(s)" and/or "processor(s)" (these terms are used interchangeably throughout the disclosure unless noted to the contrary)) **503**, a memory **529** (e.g., a read only memory (ROM) **506**, a random access memory (RAM) **505**, etc.), and/or an interface bus **507**, and most frequently, although not necessarily, are all interconnected and/or communicating through a system bus **504** on one or more (mother)board(s) **502** having conductive and/or otherwise transportive circuit pathways through which instructions (e.g., binary encoded signals) may travel to effectuate communications, operations, storage, etc. The computer systemization may be connected to a power source **586**; e.g., optionally the power source may be internal. Optionally, a cryptographic processor **526** and/or transceivers (e.g., ICs) **574** may be connected to the system bus. In another embodiment, the cryptographic processor and/or transceivers may be connected as either internal and/or external peripheral devices **512** via the interface bus I/O. In turn, the transceivers may be connected to antenna(s) **575**, thereby effectuating wireless transmission and reception of various communication and/or sensor protocols; for example the antenna(s) may connect to: a Texas Instruments WiLink WL1283 transceiver chip (e.g., providing 802.11n, Bluetooth 3.0, FM, global positioning system (GPS) (thereby allowing SMS controller to determine its location)); Broadcom BCM4329FKUBG transceiver chip (e.g., providing 802.11n, Bluetooth 2.1+EDR, FM, etc.), BCM28150 (HSPA+) and BCM2076 (Bluetooth 4.0, GPS, etc.); a Broadcom BCM4750IUB8 receiver chip (e.g., GPS); an Infineon Technologies X-Gold 618-PMB9800 (e.g., providing 2G/3G HSDPA/HSUPA communications); Intel's XMM 7160 (LTE & DC-HSPA), Qualcom's CDMA (2000), Mobile Data/Station Modem, Snapdragon; and/or the like. The system clock may have a crystal oscillator and generates a base signal through the computer systemization's circuit pathways. The clock may be coupled to the system bus and various clock multipliers that will increase or decrease the base operating frequency for other components interconnected in the computer systemization. The clock and various components in a computer systemization drive signals embodying information throughout the system. Such transmission and reception of instructions embodying information throughout a computer systemization may be referred to as communications. These communicative instructions may further be transmitted, received, and the cause of return and/or reply communications beyond the instant computer systemization to: communications networks, input devices, other computer systemizations, peripheral devices, and/or the like. It should be understood that in alternative embodiments, any of the above components may be connected directly to one another, connected to the CPU, and/or organized in numerous variations employed as exemplified by various computer systems.

[0037] The CPU comprises at least one high-speed data processor adequate to execute program components for executing user and/or system-generated requests. Often, the processors themselves will incorporate various specialized processing units, such as, but not limited to: floating point units, integer processing units, integrated system (bus) controllers, logic operating units, memory management control units, etc. and even specialized processing sub-units like graphics processing units, digital signal processing units, and/or the like. Additionally, processors may include internal fast access addressable memory, and be capable of mapping and addressing memory **529** beyond the processor itself; internal memory may include, but is not limited to: fast registers, various levels of cache memory (e.g., level 1, 2, 3, etc.), RAM, etc. The processor may access this memory through the use of a memory address space that is accessible via instruction address, which the processor can construct and decode allowing it to access a circuit path to a specific memory address space having a memory state/value. The CPU may be a microprocessor such as: AMD's Athlon, Duron and/or Opteron; ARM's classic (e.g., ARM7/9/11), embedded (Coretx-M/R), application (Cortex-A), and secure processors; IBM and/or Motorola's DragonBall and PowerPC; IBM's and Sony's Cell processor; Intel's Atom, Celeron (Mobile), Core (2/Duo/i3/i5/i7), Itanium, Pentium, Xeon, and/or XScale; and/or the like processor(s). The CPU interacts with memory through instruction passing through conductive and/or transportive conduits (e.g., (printed) electronic and/or optic circuits) to execute stored instructions (i.e., program code). Such instruction passing facilitates communication within the SMS controller and beyond through various interfaces. Should processing requirements dictate a greater amount speed and/or capacity, distributed processors (e.g., Distributed SMS), mainframe, multi-core, parallel, and/or super-computer architectures may similarly be employed. Alternatively, should deployment requirements dictate greater portability, smaller mobile devices (e.g., smartphones, Personal Digital Assistants (PDAs), etc.) may be employed.

[0038] Depending on the particular implementation, features of the SMS may be achieved by implementing a microcontroller such as CAST's R8051 XC2 microcontroller; Intel's MCS 51 (i.e., 8051 microcontroller); and/or the like. Also, to implement certain features of the SMS, some feature implementations may rely on embedded components, such as: Application-Specific Integrated Circuit ("ASIC"), Digital Signal Processing ("DSP"), Field Programmable Gate Array ("FPGA"), and/or the like embedded technology. For example, any of the SMS component collection (distributed or otherwise) and/or features may be implemented via the microprocessor and/or via embedded components; e.g., via

ASIC, coprocessor, DSP, FPGA, and/or the like. Alternately, some implementations of the SMS may be implemented with embedded components that are configured and used to achieve a variety of features or signal processing.

[0039] Depending on the particular implementation, the embedded components may include software solutions, hardware solutions, and/or some combination of both hardware/software solutions. For example, SMS features discussed herein may be achieved through implementing FPGAs, which are a semiconductor devices containing programmable logic components called "logic blocks", and programmable interconnects, such as the high performance FPGA Virtex series and/or the low cost Spartan series manufactured by Xilinx. Logic blocks and interconnects can be programmed by the customer or designer, after the FPGA is manufactured, to implement any of the SMS features. A hierarchy of programmable interconnects allow logic blocks to be interconnected as needed by the SMS system designer/administrator, somewhat like a one-chip programmable breadboard. An FPGA's logic blocks can be programmed to perform the operation of basic logic gates such as AND, and XOR, or more complex combinational operators such as decoders or simple mathematical operations. In most FPGAs, the logic blocks also include memory elements, which may be circuit flip-flops or more complete blocks of memory. In some circumstances, the SMS may be developed on regular FPGAs and then migrated into a fixed version that more resembles ASIC implementations. Alternate or coordinating implementations may migrate SMS controller features to a final ASIC instead of or in addition to FPGAs. Depending on the implementation all of the aforementioned embedded components and microprocessors may be considered the "CPU" and/or "processor" for the SMS.

Power Source

[0040] The power source **586** may be of any standard form for powering small electronic circuit board devices such as the following power cells: alkaline, lithium hydride, lithium ion, lithium polymer, nickel cadmium, solar cells, and/or the like. Other types of AC or DC power sources may be used as well. In the case of solar cells, in one embodiment, the case provides an aperture through which the solar cell may capture photonic energy. The power cell **586** is connected to at least one of the interconnected subsequent components of the SMS thereby providing an electric current to all the interconnected components. In one example, the power source **586** is connected to the system bus component **504**. In an alternative embodiment, an outside power source **586** is provided through a connection across the I/O **508** interface. For example, a USB and/or IEEE 1394 connection carries both data and power across the connection and is therefore a suitable source of power.

Interface Adapters

[0041] Interface bus(ses) **507** may accept, connect, and/or communicate to a number of interface adapters, frequently, although not necessarily in the form of adapter cards, such as but not limited to: input output interfaces (I/O) **508**, storage interfaces **509**, network interfaces **510**, and/or the like. Optionally, cryptographic processor interfaces **527** similarly may be connected to the interface bus. The interface bus provides for the communications of interface adapters with one another as well as with other components of the computer systemization. Interface adapters are adapted for a compatible interface bus. Interface adapters may connect to the interface bus via an expansion and/or slot architecture. Various expansion and/or slot architectures that be employed, such as, but not limited to: Accelerated Graphics Port (AGP), Card Bus, ExpressCard, (Extended) Industry Standard Architecture ((E)ISA), Micro Channel Architecture (MCA), NuBus, Peripheral Component Interconnect (Extended) (PCI(X)), PCI Express, Personal Computer Memory Card International Association (PCMCIA), Thunderbolt, and/or the like.

[0042] Storage interfaces **509** may accept, communicate, and/or connect to a number of storage devices such as, but not limited to: storage devices **514**, removable disc devices, and/or the like. Storage interfaces may employ connection protocols such as, but not limited to: (Ultra) (Serial) Advanced Technology Attachment (Packet Interface) ((Ultra) (Serial) ATA(PI)), (Enhanced) Integrated Drive Electronics ((E)IDE), Institute of Electrical and Electronics Engineers (IEEE) 1394, Ethernet, fiber channel, Small Computer Systems Interface (SCSI), Thunderbolt, Universal Serial Bus (USB), and/or the like.

[0043] Network interfaces **510** may accept, communicate, and/or connect to a communications network **513**. Through a communications network **513**, the SMS controller is accessible through remote clients **533***b* (e.g., computers with web browsers) by users **533***a*. Network interfaces may employ connection protocols such as, but not limited to: direct connect, Ethernet (thick, thin, twisted pair 10/100/1000 Base T, and/or the like), Token Ring, wireless connection such as IEEE 802.11a-x, and/or the like. Should processing requirements dictate a greater amount speed and/or capacity, distributed network controllers (e.g., Distributed SMS), architectures may similarly be employed to pool, load balance, and/or otherwise increase the communicative bandwidth required by the SMS controller. A communications network may be any one and/or the combination of the following: a direct interconnection; the Internet; a Local Area Network (LAN); a Metropolitan Area Network (MAN); an Operating Missions as Nodes on the Internet (OMNI); a secured custom connection; a Wide Area Network (WAN); a wireless network (e.g., employing protocols such as, but not limited to a Wireless Application Protocol (WAP), I-mode, and/or the like); and/or the like. A network interface may be regarded as a specialized form of an input output interface. Further, multiple network interfaces **510** may be used to engage with various communications network types **513**. For example, multiple network interfaces may be employed to allow for the communication over broadcast, multicast, and/or unicast networks.

[0044] Input Output interfaces (I/O) **508** may accept, communicate, and/or connect to user input devices **511**, peripheral devices **512**, cryptographic processor devices **528**, and/or the like. I/O may employ connection protocols such as, but not limited to: audio: analog, digital, monaural, RCA, stereo, and/or the like; data: Apple Desktop Bus (ADB), Bluetooth, IEEE 1394a-b, serial, universal serial bus (USB); infrared; joystick; keyboard; midi; optical; PC AT; PS/**2**; parallel; radio; video interface: Apple Desktop Connector (ADC), BNC, coaxial, component, composite, digital, DisplayPort, Digital Visual Interface (DVI), high-definition multimedia interface (HDMI), RCA, RF antennae, S-Video, VGA, and/or the like; wireless transceivers: 802.11a/b/g/n/x; Bluetooth; cellular (e.g., code division multiple access (CDMA), high speed packet access (HSPA(+)), high-speed downlink packet access (HSDPA), global system for mobile communications

(GSM), long term evolution (LTE), WiMax, etc.); and/or the like. One output device may be a video display, which may take the form of a Cathode Ray Tube (CRT), Liquid Crystal Display (LCD), Light Emitting Diode (LED), Organic Light Emitting Diode (OLED), Plasma, and/or the like based monitor with an interface (e.g., VGA, DVI circuitry and cable) that accepts signals from a video interface. The video interface composites information generated by a computer systemization and generates video signals based on the composited information in a video memory frame. Another output device is a television set, which accepts signals from a video interface. Often, the video interface provides the composited video information through a video connection interface that accepts a video display interface (e.g., an RCA composite video connector accepting an RCA composite video cable; a DVI connector accepting a DVI display cable, HDMI, etc.).

[0045] User input devices **511** often are a type of peripheral device **512** (see below) and may include: card readers, dongles, finger print readers, gloves, graphics tablets, joysticks, keyboards, microphones, mouse (mice), remote controls, retina readers, touch screens (e.g., capacitive, resistive, etc.), trackballs, trackpads, sensors (e.g., accelerometers, ambient light, GPS, gyroscopes, proximity, etc.), styluses, and/or the like.

[0046] Peripheral devices **512** may be connected and/or communicate to I/O and/or other facilities of the like such as network interfaces, storage interfaces, directly to the interface bus, system bus, the CPU, and/or the like. Peripheral devices may be external, internal and/or part of the SMS controller. Peripheral devices may include: antenna, audio devices (e.g., line-in, line-out, microphone input, speakers, etc.), cameras (e.g., still, video, webcam, etc.), dongles (e.g., for copy protection, ensuring secure transactions with a digital signature, and/or the like), external processors (for added capabilities; e.g., crypto devices **528**), force-feedback devices (e.g., vibrating motors), near field communication (NFC) devices, network interfaces, printers, radio frequency identifiers (RFIDs), scanners, storage devices, transceivers (e.g., cellular, GPS, etc.), video devices (e.g., goggles, monitors, etc.), video sources, visors, and/or the like. Peripheral devices often include types of input devices (e.g., microphones, cameras, etc.).

[0047] It should be noted that although user input devices and peripheral devices may be employed, the SMS controller may be embodied as an embedded, dedicated, and/or monitor-less (i.e., headless) device, wherein access would be provided over a network interface connection.

[0048] Cryptographic units such as, but not limited to, microcontrollers, processors **526**, interfaces **527**, and/or devices **528** may be attached, and/or communicate with the SMS controller. A MC68HC16 microcontroller, manufactured by Motorola Inc., may be used for and/or within cryptographic units. The MC68HC16 microcontroller utilizes a 16-bit multiply-and-accumulate instruction in the 16 MHz configuration and requires less than one second to perform a 512-bit RSA private key operation. Cryptographic units support the authentication of communications from interacting agents, as well as allowing for anonymous transactions. Cryptographic units may also be configured as part of the CPU. Equivalent microcontrollers and/or processors may also be used. Other commercially available specialized cryptographic processors include: the Broadcom's CryptoNetX and other Security Processors; nCipher's nShield (e.g., Solo, Connect, etc.), SafeNet's Luna PCI (e.g., 7100) series; Semaphore Communications' 40 MHz Roadrunner 184; sMIP's (e.g., 208956); Sun's Cryptographic Accelerators (e.g., Accelerator 6000 PCIe Board, Accelerator 500 Daughtercard); /(e.g., L2100, L2200, U2400) line, which is capable of performing 500+ MB/s of cryptographic instructions; VLSI Technology's 33 MHz 6868; and/or the like.

Memory

[0049] Generally, any mechanization and/or embodiment allowing a processor to affect the storage and/or retrieval of information is regarded as memory **529**. However, memory is a fungible technology and resource, thus, any number of memory embodiments may be employed in lieu of or in concert with one another. It is to be understood that the SMS controller and/or a computer systemization may employ various forms of memory **529**. For example, a computer systemization may be configured wherein the operation of on-chip CPU memory (e.g., registers), RAM, ROM, and any other storage devices are provided by a paper punch tape or paper punch card mechanism; however, such an embodiment would result in an extremely slow rate of operation. In one configuration, memory **529** will include ROM **506**, RAM **505**, and a storage device **514**. A storage device **514** may employ any number of computer storage devices/systems. Storage devices may include a drum; a (fixed and/or removable) magnetic disk drive; a magneto-optical drive; an optical drive (i.e., Blueray, CD ROM/RAM/Recordable (R)/ReWritable (RW), DVD R/RW, HD DVD R/RW etc.); an array of devices (e.g., Redundant Array of Independent Disks (RAID)); solid state memory devices (USB memory, solid state drives (SSD), etc.); other processor-readable storage mediums; and/or other devices of the like. Thus, a computer systemization generally requires and makes use of memory.

Component Collection

[0050] The memory **529** may contain a collection of program and/or database components and/or data such as, but not limited to: operating system component(s) **515** (operating system); information server component(s) **516** (information server); user interface component(s) **517** (user interface); Web browser component(s) **518** (Web browser); database(s) **519**; mail server component(s) **521**; mail client component(s) **522**; cryptographic server component(s) **520** (cryptographic server); the SMS component(s) **535**; and/or the like (i.e., collectively a component collection). These components may be stored and accessed from the storage devices and/or from storage devices accessible through an interface bus. Although non-conventional program components such as those in the component collection, may be stored in a local storage device **514**, they may also be loaded and/or stored in memory such as: peripheral devices, RAM, remote storage facilities through a communications network, ROM, various forms of memory, and/or the like.

Operating System

[0051] The operating system component **515** is an executable program component facilitating the operation of the SMS controller. The operating system may facilitate access of I/O, network interfaces, peripheral devices, storage devices, and/or the like. The operating system may be a highly fault tolerant, scalable, and secure system such as: Apple Macintosh OS X (Server); AT&T Plan 9; Be OS; Unix and Unix-like system distributions (such as AT&T's UNIX; Berkley Soft-

ware Distribution (BSD) variations such as FreeBSD, NetBSD, OpenBSD, and/or the like; Linux distributions such as Red Hat, Ubuntu, and/or the like); and/or the like operating systems. However, more limited and/or less secure operating systems also may be employed such as Apple Macintosh OS, IBM OS/2, Microsoft DOS, Microsoft Windows 2000/2003/3.1/95/98/CE/Millenium/NT/Vista/XP (Server), Palm OS, and/or the like. In addition, emobile operating systems such as Apple's iOS, Google's Android, Hewlett Packard's WebOS, Microsofts Windows Mobile, and/or the like may be employed. Any of these operating systems may be embedded within the hardware of the SMS controller, and/or stored/loaded into memory/storage. An operating system may communicate to and/or with other components in a component collection, including itself, and/or the like. Most frequently, the operating system communicates with other program components, user interfaces, and/or the like. For example, the operating system may contain, communicate, generate, obtain, and/or provide program component, system, user, and/or data communications, requests, and/or responses. The operating system, once executed by the CPU, may enable the interaction with communications networks, data, I/O, peripheral devices, program components, memory, user input devices, and/or the like. The operating system may provide communications protocols that allow the SMS controller to communicate with other entities through a communications network **513**. Various communication protocols may be used by the SMS controller as a subcarrier transport mechanism for interaction, such as, but not limited to: multicast, TCP/IP, UDP, unicast, and/or the like.

Information Server

[0052] An information server component **516** is a stored program component that is executed by a CPU. The information server may be an Internet information server such as, but not limited to Apache Software Foundation's Apache, Microsoft's Internet Information Server, and/or the like. The information server may allow for the execution of program components through facilities such as Active Server Page (ASP), ActiveX, (ANSI) (Objective-) C (++), C# and/or .NET, Common Gateway Interface (CGI) scripts, dynamic (D) hypertext markup language (HTML), FLASH, Java, JavaScript, Practical Extraction Report Language (PERL), Hypertext Pre-Processor (PHP), pipes, Python, wireless application protocol (WAP), WebObjects, and/or the like. The information server may support secure communications protocols such as, but not limited to, File Transfer Protocol (FTP); HyperText Transfer Protocol (HTTP); Secure Hypertext Transfer Protocol (HTTPS), Secure Socket Layer (SSL), messaging protocols (e.g., America Online (AOL) Instant Messenger (AIM), Apple's iMessage, Application Exchange (APEX), ICQ, Internet Relay Chat (IRC), Microsoft Network (MSN) Messenger Service, Presence and Instant Messaging Protocol (PRIM), Internet Engineering Task Force's (IETF's) Session Initiation Protocol (SIP), SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE), open XML-based Extensible Messaging and Presence Protocol (XMPP) (i.e., Jabber or Open Mobile Alliance's (OMA's) Instant Messaging and Presence Service (IMPS)), Yahoo! Instant Messenger Service, and/or the like. The information server provides results in the form of Web pages to Web browsers, and allows for the manipulated generation of the Web pages through interaction with other program components. After a Domain Name System (DNS) resolution portion of an HTTP request is resolved to a particular information server, the information server resolves requests for information at specified locations on the SMS controller based on the remainder of the HTTP request. For example, a request such as http://123.124.125.126/myInformation.html might have the IP portion of the request "123.124.125.126" resolved by a DNS server to an information server at that IP address; that information server might in turn further parse the http request for the "/myInformation.html" portion of the request and resolve it to a location in memory containing the information "myInformation.html." Additionally, other information serving protocols may be employed across various ports, e.g., FTP communications across port **21**, and/or the like. An information server may communicate to and/or with other components in a component collection, including itself, and/or facilities of the like. Most frequently, the information server communicates with the SMS database **519**, operating systems, other program components, user interfaces, Web browsers, and/or the like.

[0053] Access to the SMS database may be achieved through a number of database bridge mechanisms such as through scripting languages as enumerated below (e.g., CGI) and through inter-application communication channels as enumerated below (e.g., CORBA, WebObjects, etc.). Any data requests through a Web browser are parsed through the bridge mechanism into appropriate grammars as required by the SMS. In one embodiment, the information server would provide a Web form accessible by a Web browser. Entries made into supplied fields in the Web form are tagged as having been entered into the particular fields, and parsed as such. The entered terms are then passed along with the field tags, which act to instruct the parser to generate queries directed to appropriate tables and/or fields. In one embodiment, the parser may generate queries in standard SQL by instantiating a search string with the proper join/select commands based on the tagged text entries, wherein the resulting command is provided over the bridge mechanism to the SMS as a query. Upon generating query results from the query, the results are passed over the bridge mechanism, and may be parsed for formatting and generation of a new results Web page by the bridge mechanism. Such a new results Web page is then provided to the information server, which may supply it to the requesting Web browser.

[0054] Also, an information server may contain, communicate, generate, obtain, and/or provide program component, system, user, and/or data communications, requests, and/or responses.

User Interface

[0055] Computer interfaces in some respects are similar to automobile operation interfaces. Automobile operation interface elements such as steering wheels, gearshifts, and speedometers facilitate the access, operation, and display of automobile resources, and status. Computer interaction interface elements such as check boxes, cursors, menus, scrollers, and windows (collectively and commonly referred to as widgets) similarly facilitate the access, capabilities, operation, and display of data and computer hardware and operating system resources, and status. Operation interfaces are commonly called user interfaces. Graphical user interfaces (GUIs) such as the Apple Macintosh Operating System's Aqua and iOS's Cocoa Touch, IBM's OS/2, Google's Android Mobile UI, Microsoft's Windows 2000/2003/3.1/95/98/CE/Millenium/Mobile/NT/XP/Vista/7/8 (i.e., Aero, Metro), Unix's X-Win-

dows (e.g., which may include additional Unix graphic interface libraries and layers such as K Desktop Environment (KDE), mythTV and GNU Network Object Model Environment (GNOME)), web interface libraries (e.g., ActiveX, AJAX, (D)HTML, FLASH, Java, JavaScript, etc. interface libraries such as, but not limited to, Dojo, jQuery(UI), MooTools, Prototype, script.aculo.us, SWFObject, Yahoo! User Interface, any of which may be used and) provide a baseline and means of accessing and displaying information graphically to users.

[0056] A user interface component **517** is a stored program component that is executed by a CPU. The user interface may be a graphic user interface as provided by, with, and/or atop operating systems and/or operating environments such as already discussed. The user interface may allow for the display, execution, interaction, manipulation, and/or operation of program components and/or system facilities through textual and/or graphical facilities. The user interface provides a facility through which users may affect, interact, and/or operate a computer system. A user interface may communicate to and/or with other components in a component collection, including itself, and/or facilities of the like. Most frequently, the user interface communicates with operating systems, other program components, and/or the like. The user interface may contain, communicate, generate, obtain, and/or provide program component, system, user, and/or data communications, requests, and/or responses.

Web Browser

[0057] A Web browser component **518** is a stored program component that is executed by a CPU. The Web browser may be a hypertext viewing application such as Google's (Mobile) Chrome, Microsoft Internet Explorer, Netscape Navigator, Apple's (Mobile) Safari, embedded web browser objects such as through Apple's Cocoa (Touch) object class, and/or the like. Secure Web browsing may be supplied with 128 bit (or greater) encryption by way of HTTPS, SSL, and/or the like. Web browsers allowing for the execution of program components through facilities such as ActiveX, AJAX, (D)HTML, FLASH, Java, JavaScript, web browser plug-in APIs (e.g., Chrome, FireFox, Internet Explorer, Safari Plug-in, and/or the like APIs), and/or the like. Web browsers and like information access tools may be integrated into PDAs, cellular telephones, smartphones, and/or other mobile devices. A Web browser may communicate to and/or with other components in a component collection, including itself, and/or facilities of the like. Most frequently, the Web browser communicates with information servers, operating systems, integrated program components (e.g., plug-ins), and/or the like; e.g., it may contain, communicate, generate, obtain, and/or provide program component, system, user, and/or data communications, requests, and/or responses. Also, in place of a Web browser and information server, a combined application may be developed to perform similar operations of both. The combined application would similarly effect the obtaining and the provision of information to users, user agents, and/or the like from the SMS equipped nodes. The combined application may be nugatory on systems employing standard Web browsers.

Mail Server

[0058] A mail server component **521** is a stored program component that is executed by a CPU **503**. The mail server may be an Internet mail server such as, but not limited to Apple's Mail Server (3), dovecot, sendmail, Microsoft Exchange, and/or the like. The mail server may allow for the execution of program components through facilities such as ASP, ActiveX, (ANSI) (Objective-) C (++), C# and/or .NET, CGI scripts, Java, JavaScript, PERL, PHP, pipes, Python, WebObjects, and/or the like. The mail server may support communications protocols such as, but not limited to: Internet message access protocol (IMAP), Messaging Application Programming Interface (MAPI)/Microsoft Exchange, post office protocol (POP3), simple mail transfer protocol (SMTP), and/or the like. The mail server can route, forward, and process incoming and outgoing mail messages that have been sent, relayed and/or otherwise traversing through and/or to the SMS.

[0059] Access to the SMS mail may be achieved through a number of APIs offered by the individual Web server components and/or the operating system.

[0060] Also, a mail server may contain, communicate, generate, obtain, and/or provide program component, system, user, and/or data communications, requests, information, and/or responses.

Mail Client

[0061] A mail client component **522** is a stored program component that is executed by a CPU **503**. The mail client may be a mail viewing application such as Apple (Mobile) Mail, Microsoft Entourage, Microsoft Outlook, Microsoft Outlook Express, Mozilla, Thunderbird, and/or the like. Mail clients may support a number of transfer protocols, such as: IMAP, Microsoft Exchange, POP3, SMTP, and/or the like. A mail client may communicate to and/or with other components in a component collection, including itself, and/or facilities of the like. Most frequently, the mail client communicates with mail servers, operating systems, other mail clients, and/or the like; e.g., it may contain, communicate, generate, obtain, and/or provide program component, system, user, and/or data communications, requests, information, and/or responses. Generally, the mail client provides a facility to compose and transmit electronic mail messages.

Cryptographic Server

[0062] A cryptographic server component **520** is a stored program component that is executed by a CPU **503**, cryptographic processor **526**, cryptographic processor interface **527**, cryptographic processor device **528**, and/or the like. Cryptographic processor interfaces will allow for expedition of encryption and/or decryption requests by the cryptographic component; however, the cryptographic component, alternatively, may run on a CPU. The cryptographic component allows for the encryption and/or decryption of provided data. The cryptographic component allows for both symmetric and asymmetric (e.g., Pretty Good Protection (PGP)) encryption and/or decryption. The cryptographic component may employ cryptographic techniques such as, but not limited to: digital certificates (e.g., X.509 authentication framework), digital signatures, dual signatures, enveloping, password access protection, public key management, and/or the like. The cryptographic component will facilitate numerous (encryption and/or decryption) security protocols such as, but not limited to: checksum, Data Encryption Standard (DES), Elliptical Curve Encryption (ECC), International Data Encryption Algorithm (IDEA), Message Digest 5 (MD5,

which is a one way hash operation), passwords, Rivest Cipher (RC5), Rijndael, RSA (which is an Internet encryption and authentication system that uses an algorithm developed in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman), Secure Hash Algorithm (SHA), Secure Socket Layer (SSL), Secure Hypertext Transfer Protocol (HTTPS), and/or the like. Employing such encryption security protocols, the SMS may encrypt all incoming and/or outgoing communications and may serve as node within a virtual private network (VPN) with a wider communications network. The cryptographic component facilitates the process of "security authorization" whereby access to a resource is inhibited by a security protocol wherein the cryptographic component effects authorized access to the secured resource. In addition, the cryptographic component may provide unique identifiers of content, e.g., employing and MD5 hash to obtain a unique signature for an digital audio file. A cryptographic component may communicate to and/or with other components in a component collection, including itself, and/or facilities of the like. The cryptographic component supports encryption schemes allowing for the secure transmission of information across a communications network to enable the SMS component to engage in secure transactions if so desired. The cryptographic component facilitates the secure accessing of resources on the SMS and facilitates the access of secured resources on remote systems; i.e., it may act as a client and/or server of secured resources. Most frequently, the cryptographic component communicates with information servers, operating systems, other program components, and/or the like. The cryptographic component may contain, communicate, generate, obtain, and/or provide program component, system, user, and/or data communications, requests, and/or responses.

The SMS Database

[0063] The SMS database component **519** may be embodied in a database and its stored data. The database is a stored program component, which is executed by the CPU; the stored program component portion configuring the CPU to process the stored data. The database may be any of a number of fault tolerant, relational, scalable, secure database such as DB2, MySQL, Oracle, Sybase, and/or the like. Relational databases are an extension of a flat file. Relational databases consist of a series of related tables. The tables are interconnected via a key field. Use of the key field allows the combination of the tables by indexing against the key field; i.e., the key fields act as dimensional pivot points for combining information from various tables. Relationships generally identify links maintained between tables by matching primary keys. Primary keys represent fields that uniquely identify the rows of a table in a relational database. More precisely, they uniquely identify rows of a table on the "one" side of a one-to-many relationship.

[0064] Alternatively, the SMS database may be implemented using various standard data-structures, such as an array, hash, (linked) list, struct, structured text file (e.g., XML), table, and/or the like. Such data-structures may be stored in memory and/or in (structured) files. In another alternative, an object-oriented database may be used, such as Frontier, ObjectStore, Poet, Zope, and/or the like. Object databases can include a number of object collections that are grouped and/or linked together by common attributes; they may be related to other object collections by some common attributes. Object-oriented databases perform similarly to relational databases with the exception that objects are not just pieces of data but may have other types of capabilities encapsulated within a given object. If the SMS database is implemented as a data-structure, the use of the SMS database **519** may be integrated into another component such as the SMS component **535**. Also, the database may be implemented as a mix of data structures, objects, and relational structures. Databases may be consolidated and/or distributed in countless variations through standard data processing techniques. Portions of databases, e.g., tables, may be exported and/or imported and thus decentralized and/or integrated.

[0065] In one embodiment, the database component **519** includes several tables **519***a-l*. A Users table **519***a* may include fields such as, but not limited to: user_id, ssn, dob, first_name, last_name, age, state, address_firstline, address_secondline, zipcode, devices_list, contact_info, contact_type, alt_contact_info, alt_contact_type, user_biometrics, and/or the like. The Users table may support and/or track multiple entity accounts on a SMS. A Devices table **519***b* may include fields such as, but not limited to: device_ID, device_name, device_IP, device_MAC, device_type, device_model, device_version, device_OS, device_apps list, device_securekey, wallet_app_installed_flag, device_browser, device_plugin_list, device_font_list, device_screen_size, device_time_zone, and/or the like. An Apps table **519***c* may include fields such as, but not limited to: app_ID, app_name, app_type, app_dependencies, and/or the like. An Accounts table **519***d* may include fields such as, but not limited to: account_number, account_security_code, account_name, issuer_acquirer_flag, issuer_name, acquirer_name, account_address, routing_number, access_API_call, linked_wallets_list, and/or the like. A Merchants table **519***e* may include fields such as, but not limited to: merchant_id, merchant_name, merchant_address, ip_address, mac_address, auth_key, port_num, security_settings_list, and/or the like. An Issuers table **519***f* may include fields such as, but not limited to: issuer_id, issuer_name, issuer_address, ip_address, mac_address, auth_key, port_num, security_settings_list, and/or the like. An Acquirers table **519***g* may include fields such as, but not limited to: acquirer_id, account_firstname, account_lastname, account_type, account_num, account_balance_list, billingaddress_line1, billingaddress_line2, billing_zipcode, billing_state, shipping_preferences, shippingaddress_line1, shippingaddress_line2, shipping_zipcode, shipping_state, and/or the like. A Pay Gateways table **519***h* may include fields such as, but not limited to: gateway_ID, gateway_IP, gateway_MAC, gateway_secure_key, gateway_access_list, gateway_API_call_list, gateway_services_list, and/or the like. A Transactions table **519***i* may include fields such as, but not limited to: order_id, user_id, timestamp, transaction_cost, purchase_details_list, num_products, products_list, product_type, product_params_list, product_title, product_summary, quantity, user_id, client_id, client_ip, client_type, client_model, operating_system, os_version, app_installed_flag, user_id, account_firstname, account_lastname, account_type, account_num, account_priority_account_ratio, billingaddress_line1, billingaddress_line2, billing_zipcode, billing_state, shipping_preferences, shippingaddress_line1, shippingaddress_line2, shipping_zipcode, shipping_state, merchant_id, merchant_name, merchant_auth_key, and/or the like. A Batches table **519***j* may include fields such as, but not limited to: batch_id, transaction_id_list, timestamp_list, cleared_flag_list, clearance_trigger_settings, and/or the like. A Decoded Code Contents table **519***k* may include fields such as, but not limited to: code_id, timestamp, link_id,

app_id, scripts_id, links_blacklist, links_whitelist, apps_blacklist, links_whitelist, and/or the like. A Products table **519***l* may include fields such as, but not limited to: product_ID, product_title, product_attributes_list, product_price, tax_info_list, related_products_list, offers_list, discounts_list, rewards_list, merchants_list, merchant_availability_list, and/or the like. A Digital Signatures table **519***m* may include fields such as, but not limited to: digital_sign_ID, digital_sign_whitelist, digital_sign_blacklist, plugins_list, fonts_list, time_zones, screen_size, flash_id, user_agent_id, and/or the like.

[0066] In one embodiment, the SMS database may interact with other database systems. For example, employing a distributed database system, queries and data access by search SMS component may treat the combination of the SMS database, an integrated data security layer database as a single database entity.

[0067] In one embodiment, user programs may contain various user interface primitives, which may serve to update the SMS. Also, various accounts may require custom database tables depending upon the environments and the types of clients the SMS may need to serve. It should be noted that any unique fields may be designated as a key field throughout. In an alternative embodiment, these tables have been decentralized into their own databases and their respective database controllers (i.e., individual database controllers for each of the above tables). Employing standard data processing techniques, one may further distribute the databases over several computer systemizations and/or storage devices. Similarly, configurations of the decentralized database controllers may be varied by consolidating and/or distributing the various database components **519***a-l*. The SMS may be configured to keep track of various settings, inputs, and parameters via database controllers.

[0068] The SMS database may communicate to and/or with other components in a component collection, including itself, and/or facilities of the like. Most frequently, the SMS database communicates with the SMS component, other program components, and/or the like. The database may contain, retain, and provide information regarding other nodes and data.

The SMSs

[0069] The SMS component **535** is a stored program component that is executed by a CPU. In one embodiment, the SMS component incorporates any and/or all combinations of the aspects of the SMS discussed in the previous figures. As such, the SMS affects accessing, obtaining and the provision of information, services, transactions, and/or the like across various communications networks.

[0070] The SMS component may provide verification, access and security, via SMS components, to virtual wallet based electronic financial transactions. In one embodiment, the SMS component **535** takes inputs (e.g., code snapshot input **109**; security verification request in; purchase checkout request **202**; wallet access authorization request **204**; and/or the like) etc., and transforms the inputs via various components (e.g., DCDV **523**; DCPV **524**; UIV **525**; and/or the like), into outputs (e.g., verified code contents **117**; purchase checkout response **203**; wallet authorization **210**; and/or the like).

[0071] The SMS component enabling access of information between nodes may be developed by employing standard development tools and languages such as, but not limited to: Apache components, Assembly, ActiveX, binary executables, (ANSI) (Objective-) C (++), C# and/or .NET, database adapters, CGI scripts, Java, JavaScript, mapping tools, procedural and object oriented development tools, PERL, PHP, Python, shell scripts, SQL commands, web application server extensions, web development environments and libraries (e.g., Microsoft's ActiveX; Adobe AIR, FLEX & FLASH; AJAX; (D)HTML; Dojo, Java; JavaScript; jQuery(UI); MooTools; Prototype; script.aculo.us; Simple Object Access Protocol (SOAP); SWFObject; Yahoo! User Interface; and/or the like), WebObjects, and/or the like. In one embodiment, the SMS server employs a cryptographic server to encrypt and decrypt communications. The SMS component may communicate to and/or with other components in a component collection, including itself, and/or facilities of the like. Most frequently, the SMS component communicates with the SMS database, operating systems, other program components, and/or the like. The SMS may contain, communicate, generate, obtain, and/or provide program component, system, user, and/or data communications, requests, and/or responses.

Distributed SMSs

[0072] The structure and/or operation of any of the SMS node controller components may be combined, consolidated, and/or distributed in any number of ways to facilitate development and/or deployment. Similarly, the component collection may be combined in any number of ways to facilitate deployment and/or development. To accomplish this, one may integrate the components into a common code base or in a facility that can dynamically load the components on demand in an integrated fashion.

[0073] The component collection may be consolidated and/or distributed in countless variations through standard data processing and/or development techniques. Multiple instances of any one of the program components in the program component collection may be instantiated on a single node, and/or across numerous nodes to improve performance through load-balancing and/or data-processing techniques. Furthermore, single instances may also be distributed across multiple controllers and/or storage devices; e.g., databases. All program component instances and controllers working in concert may do so through standard data processing communication techniques.

[0074] The configuration of the SMS controller will depend on the context of system deployment. Factors such as, but not limited to, the budget, capacity, location, and/or use of the underlying hardware resources may affect deployment requirements and configuration. Regardless of if the configuration results in more consolidated and/or integrated program components, results in a more distributed series of program components, and/or results in some combination between a consolidated and distributed configuration, data may be communicated, obtained, and/or provided. Instances of components consolidated into a common code base from the program component collection may communicate, obtain, and/or provide data. This may be accomplished through intra-application data processing communication techniques such as, but not limited to: data referencing (e.g., pointers), internal messaging, object instance variable communication, shared memory space, variable passing, and/or the like.

[0075] If component collection components are discrete, separate, and/or external to one another, then communicating, obtaining, and/or providing data with and/or to other components may be accomplished through inter-application data

processing communication techniques such as, but not limited to: Application Program Interfaces (API) information passage; (distributed) Component Object Model ((D)COM), (Distributed) Object Linking and Embedding ((D)OLE), and/or the like), Common Object Request Broker Architecture (CORBA), Jini local and remote application program interfaces, JavaScript Object Notation (JSON), Remote Method Invocation (RMI), SOAP, process pipes, shared files, and/or the like. Messages sent between discrete component components for inter-application communication or within memory spaces of a singular component for intra-application communication may be facilitated through the creation and parsing of a grammar. A grammar may be developed by using development tools such as lex, yacc, XML, and/or the like, which allow for grammar generation and parsing capabilities, which in turn may form the basis of communication messages within and between components.

[0076] For example, a grammar may be arranged to recognize the tokens of an HTTP post command, e.g.:

[0077] w3c-post http:// . . . Value1

[0078] where Value1 is discerned as being a parameter because "http://" is part of the grammar syntax, and what follows is considered part of the post value. Similarly, with such a grammar, a variable "Value1" may be inserted into an "http://" post command and then sent. The grammar syntax itself may be presented as structured data that is interpreted and/or otherwise used to generate the parsing mechanism (e.g., a syntax description text file as processed by lex, yacc, etc.). Also, once the parsing mechanism is generated and/or instantiated, it itself may process and/or parse structured data such as, but not limited to: character (e.g., tab) delineated text, HTML, structured text streams, XML, and/or the like structured data. In another embodiment, inter-application data processing protocols themselves may have integrated and/or readily available parsers (e.g., JSON, SOAP, and/or like parsers) that may be employed to parse (e.g., communications) data. Further, the parsing grammar may be used beyond message parsing, but may also be used to parse: databases, data collections, data stores, structured data, and/or the like. Again, the desired configuration will depend upon the context, environment, and requirements of system deployment.

[0079] For example, in some implementations, the SMS controller may be executing a PHP script implementing a Secure Sockets Layer ("SSL") socket server via the information server, which listens to incoming communications on a server port to which a client may send data, e.g., data encoded in JSON format. Upon identifying an incoming communication, the PHP script may read the incoming message from the client device, parse the received JSON-encoded text data to extract information from the JSON-encoded text data into PHP script variables, and store the data (e.g., client identifying information, etc.) and/or extracted information in a relational database accessible using the Structured Query Language ("SQL"). An exemplary listing, written substantially in the form of PHP/SQL commands, to accept JSON-encoded input data from a client device via a SSL connection, parse the data to extract variables, and store the data to a database, is provided below:

```
<?PHP
header('Content-Type: text/plain');
// set ip address and port to listen to for incoming data
$address = '192.168.0.100';
$port = 255;
// create a server-side SSL socket, listen for/accept incoming
communication
$sock = socket_create(AF_INET, SOCK_STREAM, 0);
socket_bind($sock, $address, $port) or die('Could not bind to address');
socket_listen($sock);
$client = socket_accept($sock);
// read input data from client device in 1024 byte blocks until end of
message
do {
    $input = "";
    $input = socket_read($client, 1024);
    $data .= $input;
} while($input != "");
// parse data to extract variables
$obj = json_decode($data, true);
// store input data in a database
mysql_connect("201.408.185.132",$DBserver,$password); // access
database server
mysql_select("CLIENT_DB.SQL"); // select database to append
mysql_query("INSERT INTO UserTable (transmission)
VALUES ($data)"); // add data to UserTable table in a CLIENT database
mysql_close("CLIENT_DB.SQL"); // close connection to database
?>
```

[0080] Also, the following resources may be used to provide example embodiments regarding SOAP parser implementation:

```
http://www.xav.com/perl/site/lib/SOAP/Parser.html
http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm
    .IBMDI.doc/referenceguide295.htm
```

[0081] and other parser implementations:

```
http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm
    .IBMDI.doc/referenceguide259.htm
```

[0082] all of which are hereby expressly incorporated by reference herein.

[0083] In order to address various issues and advance the art, the entirety of this application for SNAP MOBILE SECURITY APPARATUSES, METHODS AND SYSTEMS (including the Cover Page, Title, Headings, Field, Background, Summary, Brief Description of the Drawings, Detailed Description, Claims, Abstract, Figures, Appendices and/or otherwise) shows, by way of illustration, various example embodiments in which the claimed innovations may be practiced. The advantages and features of the application are of a representative sample of embodiments only, and are not exhaustive and/or exclusive. They are presented only to

assist in understanding and teach the claimed principles. It should be understood that they are not representative of all claimed innovations. As such, certain aspects of the disclosure have not been discussed herein. That alternate embodiments may not have been presented for a specific portion of the innovations or that further undescribed alternate embodiments may be available for a portion is not to be considered a disclaimer of those alternate embodiments. It will be appreciated that many of those undescribed embodiments incorporate the same principles of the innovations and others are equivalent. Thus, it is to be understood that other embodiments may be utilized and functional, logical, operational, organizational, structural and/or topological modifications may be made without departing from the scope and/or spirit of the disclosure. As such, all examples and/or embodiments are deemed to be non-limiting throughout this disclosure. Also, no inference should be drawn regarding those embodiments discussed herein relative to those not discussed herein other than it is as such for purposes of reducing space and repetition. For instance, it is to be understood that the logical and/or topological structure of any combination of any data flow sequence(s), program components (a component collection), other components, and/or any present feature sets as described in the figures and/or throughout are not limited to a fixed operating order and/or arrangement, but rather, any disclosed order is exemplary and all equivalents, regardless of order, are contemplated by the disclosure. Furthermore, it is to be understood that such features are not limited to serial execution, but rather, any number of threads, processes, processors, services, servers, and/or the like that may execute asynchronously, concurrently, in parallel, simultaneously, synchronously, and/or the like also are contemplated by the disclosure. As such, some of these features may be mutually contradictory, in that they cannot be simultaneously present in a single embodiment. Similarly, some features are applicable to one aspect of the innovations, and inapplicable to others. In addition, the disclosure includes other innovations not presently claimed. Applicant reserves all rights in those presently unclaimed innovations, including the right to claim such innovations, file additional applications, continuations, continuations-in-part, divisions, and/or the like thereof. As such, it should be understood that advantages, embodiments, examples, functional, features, logical, operational, organizational, structural, topological, and/or other aspects of the disclosure are not to be considered limitations on the disclosure as defined by the claims or limitations on equivalents to the claims. It is to be understood that, depending on the particular needs and/or characteristics of a SMS individual and/or enterprise user, database configuration and/or relational model, data type, data transmission and/or network framework, syntax structure, and/or the like, various embodiments of the SMS may be implemented that allow a great deal of flexibility and customization. For example, aspects of the SMS may be adapted for securing online shopping, information exchange and processing, and/or the like. While various embodiments and discussions of the SMS have been directed to electronic purchase transactions, however, it is to be understood that the embodiments described herein may be readily configured and/or customized for a wide variety of other applications and/or implementations.

What is claimed is:

1. A snap mobile security processor-implemented method, comprising:

receiving, through one or more processors, a request from a user's device to decode a scannable code and verify the security of the decoded code's contents;

decoding, through the one or more processors, the scannable code to obtain code contents requesting access to the wallet account;

obtaining from the user, through the one or more processors, digital fingerprints of the user device and a request identifier for the request to access the wallet account;

receiving from the access requester, through the one or more processors, digital signatures for the requester and the request identifier;

confirming, through the one or more processors, that the digital fingerprints of the user device verify the device is authorized to access the wallet account;

confirming, through the one or more processors, the received digital signatures verify the access requester and the request are authorized to access the wallet account; and

sending, through the one or more processors, wallet unlock and activity unlock keys to the device for activity access to the wallet.

2. The method of claim **1**, wherein the scannable code is provided by receiving a snapshot of a quick response (QR) code.

3. The method of claim **2**, wherein the wireless mobile communication device is used to capture an image of the QR code.

4. The method of claim **2**, wherein the undecoded snapshot is transferred to a security server for decoding the code contents.

5. The system of claim **4**, wherein the security server decodes the undecoded code and verifies validity and security of the contents of the decoded code.

6. The method of claim **5**, wherein contents of the decoded code contain a Uniform Resource Locator (URL) link that leads to a fraudulent website.

7. The method of claim **5**, wherein the decoded code may compromises security of a user's device by subjecting it to malicious attacks.

8. The method of claim **1**, wherein the code contents of the decoded code is sent to a security server to verify the validity and security of the contents.

9. The method of claim **1**, wherein the contents of the decoded code is determined to pose a security risk to the user device by comparing the decoded code contents to black and white lists of the code contents.

10. The method of claim **1**, wherein a user's wireless mobile communications device redirected to a link for the wireless mobile communications device to execute the link.

11. The method of claim **1**, wherein based upon retrieved links being determined in the decoded code as secure, a user's wireless mobile communications device launches one or more of the retrieved links.

12. The method of claim **11**, wherein a web hosting server responds back to a security server with the requested destination of one of the retrieved links.

13. The method of claim **12**, wherein the web hosting server provides a webpage in response to the security server.

14. The method of claim **13**, wherein contents of the webpage are provided to the user's wireless mobile communications device.

15. The method of claim **1**, wherein based upon verifying security of the decoded contents of the code, the user's wireless mobile communications device executes on the decoded contents.

16. The method of claim **15**, wherein the codes include item codes for products.

17. The method of claim **16**, wherein the user's wireless mobile communications device launches a URL link to initiate a purchase request.

18. The method of claim **16**, wherein the user's wireless mobile communications device downloads and launches an application to initiate a purchase request.

19. The method of claim **16**, wherein a webpage at the URL link requests access to a wallet account on the user's wireless mobile communications device to initiate payment for the purchase.

20. The method of claim **16**, wherein a webpage at the URL link access to a wallet account on the user's wireless mobile communications device to initiate payment for the purchase.

21. The method of claim **16**, wherein a launched app requests access to a wallet account on the user's wireless mobile communications device to initiate payment for the purchase.

22. A snap mobile security system, comprising:

a processor; and

a memory disposed in communication with the processor and storing processor-issuable instructions to:

receive a request from a user's device to decode a scannable code and verify the security of the decoded code's contents;

decode the scannable code to obtain code contents requesting access to the wallet account;

obtain from the user digital fingerprints of the user device and a request identifier for the request to access the wallet account;

receive from the access requester digital signatures for the requester and the request identifier;

confirm the digital fingerprints of the user device verify the device is authorized to access the wallet account;

confirm the received digital signatures verify the access requester and the request are authorized to access the wallet account; and

send wallet unlock and activity unlock keys to the device for activity access to the wallet.

23. A processor-readable non-transitory medium storing processor-issuable snap mobile security instructions to:

receive a request from a user's device to decode a scannable code and verify the security of the decoded code's contents;

decode the scannable code to obtain code contents requesting access to the wallet account;

obtain from the user digital fingerprints of the user device and a request identifier for the request to access the wallet account;

receive from the access requester digital signatures for the requester and the request identifier;

confirm the digital fingerprints of the user device verify the device is authorized to access the wallet account;

confirm the received digital signatures verify the access requester and the request are authorized to access the wallet account; and

send wallet unlock and activity unlock keys to the device for activity access to the wallet.

* * * * *