A new patented methodology for implementing cryptography in order to increase the security of data.

Capzul

○ *indicates a schematic*

INTRODUCTION

BACKGROUND

Data security is of paramount importance as more and more data is collected and maintained in network-based systems. An important component of this security is securing communications between devices. Specifically, huge amounts of data are constantly exchanged among network-connected devices. Exchanges are in the form of messages, documents and other data communicated between devices, including emails, attachments, instant messages, files and others.

Today, a large volume of data including but not limited to emails, documents and other files are communicated with either no security or limited and ineffective security, meaning that such communications can be readily intercepted and misappropriated by malicious, ill-intentioned third-party devices.

While mechanisms exist for securing such communications, they can be complex to implement and burdensome to operate, and as has been proven, they can frequently be compromised. For example, most existing systems for securing communications rely on asymmetric encryption methods where a publicly available key is used to encrypt data and a private key is used to decrypt it. Asymmetric encryption methods are problematic in that they do not offer as strong a protection as symmetric ones. Moreover, since the public/private key pair remains unchanged, once a key pair is compromised, a vast amount of communications can be deciphered.

Symmetric encryption methods also exist for securing communications that offer stronger protection than asymmetric methods. However, such methods are cumbersome to use. For example, they typically involve exchanging keys out of band, making the setting or renewal of keys cumbersome and therefore, infrequently effected. Accordingly, there is a need for a system and method for a secure communications that both affords much stronger protection and is convenient to use.

CURRENT "STATE OF THE ART" OF DATA TRANSMISSION CRYPTOGRAPHY

The use of encryption is essential to providing even a minimum level of security for data transmitted or shared over the Internet.

Whatever the origin, destination or type of data (e.g. text/audio/video), the use of encryption is required to protect this data against interception during transmission.

There are several symmetric cryptographic algorithms that achieve an extremely high level of protection. However, these are precluded from being used in many everyday cases because the encryption key cannot be shared.

For example, access can realistically only be granted to a Home Banking client or a customer shopping in an on-line store with the use of asymmetric cryptography. There is no way for the private cryptographic key to be shared between the client/customer and the provider without security being compromised.

The effective use of symmetric encryption is only possible when the same person(s)/system(s) of the same hierarchical level both encrypts and decrypts, for example, a single User encrypting a file to be stored in an otherwise insecure cloud storage system. Consequently, only that User would be able to decrypt and use the file.

Currently, in the great majority of cases where data is exchanged between different users (or servers or systems), the use of asymmetric encryption is required. Asymmetric encryption is the only cryptographic solution that enables the exchange of encrypted messages between two or more users, without one knowing the private cryptographic key of the other.

Asymmetric encryption was a major evolution in securing data, during transmission or while in storage. A major challenge/problem is that no mathematics exists to prove the extent to which any asymmetric cryptography algorithm is in fact safe, or safer than others. Evaluation of, and conclusion about security levels can only be achieved empirically. These conclusions of the security level are based on the difficulty of, and the computational time that is required to perform complex calculations. Consequently, the optimal way to try to improve the security of any implementation that uses asymmetric cryptography has been the increase of the public cryptographic key size.

There are a small number of asymmetric cryptographic algorithms that are in use. It is extremely difficult to develop a new one, and even more difficult to prove that it is a better and safer algorithm than the ones in use.

RSA is by far the most widely used asymmetric encryption worldwide. Arguably, the main reason for this is the cumulative effect, and the respectability gained as a result of approximately 40 years of testing. Although several vulnerabilities have been found in RSA, none of these has completely compromised the solution.

It is also thought that there are other vulnerabilities, ones that are kept secret by, for example groups working for government intelligences agencies, and equally likely, by others.

Despite these acknowledged and suspected vulnerabilities, the principal corrective undertaken during the RSA-use era to combat its vulnerabilities was to increase the size of the keys. This is done to offset the inevitable computational gains made by the development of technology and new methods of factoring very large numbers, which have been used as weapons by hackers.

Asymmetric cryptography solutions that are in use today are static, and it is safe to assume that the vast majority of algorithms are open and known, as is the demand of the marketplace. Discovering and gaining acceptance for new asymmetrical algorithms is itself an extremely difficult, complex task. The current practice of using open and known algorithms would somewhat mitigate the potential gains of a new algorithm, as its openness would increase its vulnerability to attacks.

The protection strength of the most popular asymmetric encryption algorithm, RSA, and the Diffie Hellman key exchange method are premised on the difficulty in factoring big numbers and the difficulty in calculating discreet logarithms; any new protection based on this philosophy will potentially only increase the difficulty, but would essentially be the same in terms of the target and the attack method.

The current state of IT security levels demands a solution such as CAPZUL, a methodology of open and known algorithms that when implemented, provides levels of security attained by symmetric cryptography even when asymmetric cryptography is incorporated in the solution.

Ultimately one can only expect to attain greater protection for mobile data with the introduction of a new solution / methodology for implementing asymmetric cryptography.

FAILURES, VULNERABILITIES OF CURRENTLY USED CRYPTOGRAPHY

An inherent issue with RSA and other asymmetric cryptography is that one needs to encrypt in a way that the key to decrypt cannot be the same as the key to encrypt. There must be a mathematic formula that relates the two cryptographic keys. In order to provide the greatest level of security, this formula has to be such that to extract the private key from the public key requires complex mathematic calculation, in a way that is not possible with the current, known technology.

Unfortunately, as history has proven, it inevitably becomes possible to effect this extraction when new, faster, technology is developed.

The creation of this mathematical formula is entirely empirical. There is no exact process that leads to the creation of a formula with the level of efficiency that is desired. As a consequence, it is virtually impossible to arrive at a completely accurate conclusion as to which formula is best in terms of resistance to attack.

To state the obvious, encryption only creates protection for data for the period between the encryption of the data and its decryption; that is, during transport or storage of encrypted data.

There are basically three ways to attack encrypted data by brute force:

1. Try all the possible combinations to get the desired result, which is the discovery of the key. It is difficult to ascertain the time required for these attacks to succeed – determining factors include speed of machine, number of machines, algorithms used for acceleration of factoring the numbers – and good fortune. It is essential to recognize that all encryptions are vulnerable to these types of attacks. The extent to which these attacks are successful is determined by the quality of the cryptographic algorithm.

   This method can be used to try to attack both symmetric and asymmetric encryption

2. Try to discover the private key from the public key, given that the public key is known. As an example, in the case of RSA one would get the private key by factoring the public key thus discovering the two prime numbers that were used to generate the keys.

This method can only be used for attacking asymmetric encryption.

3. Try to discover the data from the public key, without using the private key. This is a less explored method of attack, but one that nevertheless needs to be considered as a distinctly realistic possibility.

This method can only be used for attacking asymmetric encryption.

There are several methods of attack other then those perpetrated with the use of brute force that can be used against both symmetric and asymmetric encryption.

Several attacks are possible, predicated on, and determined by specific weaknesses and tendencies of the algorithms involved. The more random is the result of encryption of the data, the more difficult it is to find such weaknesses.

Attack using reverse engineering can be initiated in various ways. An example of a basic one involves analyzing the memory (automatically or manually) when the program that implements the cryptography is running. This type of attack, when successful, will result in the discovery of the targeted cryptographic key.

CAPZUL

A NEW METHODOLOGY FOR IMPLEMENTING EXISTING CRYPTOGRAPHY

CAPZUL is a new security solution that exponentially elevates the security levels of applications, services and solutions with which it is used. It provides a level of security for asymmetric cryptography previously only available in symmetric cryptography – critical if mobile data is to be truly secure.

The four basic components of CAPZUL are as follows:

1. A Patent Pending new methodology of using asymmetric encryption. All forms of attacks mentioned above (other than attacks based on reverse engineering) are avoided or made exponentially more difficult with this new encryption methodology.

   It is essential to note that no new algorithms are being introduced with CAPZUL; standard industry cryptography is implemented with a far more secure methodology.

2. Various protections against reverse engineering-based attacks have been implemented. These protections, rarely used or implemented effectively elsewhere, are essential, as there is no mathematic protection to thwart this type of attack at the end points. No encryption itself is immune to such attack. Several ways of protection against reverse engineering in the implementation of any solution have to be created in order to ensure success in warding off these attacks. These protections are not related to cryptographic algorithms or methodologies. CAPZUL does this to complement the protection provided with the new methodology of using cryptographies, because reverse engineering attacks are ultimately the most common way to break the security of a computer system.

3. CAPZUL implements the new methodology and protection against reverse engineering in a monolithic way. It makes identifying software layers or software interfaces extremely difficult. If the solution is not monolithic, reverse engineering attacks pose much greater danger and are, ultimately, effective.

4. APIs that facilitate the seamless integration of the new methodology with any application, solution or service that requires heightened data security.

KEY ELEMENTS OF THE NEW CRYPTOGRAPHIC METHODOLOGY

1. Several keys, both symmetric and asymmetric, are used. In most cases these keys are independent of each other. Because of this independence, in the highly unlikely event that an attack successfully breaks one key, the others still protect the data.

2. High frequency of generation of asymmetric and symmetric cryptographic keys.

   Because the exchange of the public, private and symmetric keys is very frequent for each User, in the most unlikely of situation that a group of keys were actually to be broken, only one message would be compromised. Nothing will have been gained so as to provide the intercepting party with any insight that would facilitate further compromises, as each subsequent encryption would use a new set of keys.

3. Automatic certification.

   After the generation of the first key, when the User is accepted by the system, the certification of new keys is already part of mathematics implemented by the new methodology.

4. Airtight synchronization between system Users.

   When the first exchange of messages between two Users starts, there is an exchange of information that allows them to establish and then maintain the synchronization with each other. It is crucial to understand the implication here; unless all the keys of all historical communication between these Users are broken and available (and this is only available to the two Users), there will not be enough information to decrypt or encrypt the next message. In addition to creating a much stronger cryptographic protection than could be achievable without this synchronization, this provides the highest level of certainty of the source and destination of the message (or encrypted data).

5. The simultaneous existence of multiple public keys for each User.

Anyone attacking the system has no way to predict which of the multiple public keys will be the next public key to be used, since there may be thousands valid public keys at all times from which the public key to be used is chosen.

6. Ability to use a public key unique for each message and User.

This means that essentially, every message (data packet) has a new (different) public key. In using a new public key for every message, there is a situation created where **the public key is in fact secret.** Only those who will use it, at a given precise moment, need to, and will know this public key. The clear implication of a secret public key is that all specific attacks against asymmetric encryption are unfeasible, because in order to initiate such attacks the prerequisite need is to know the public key.

7. The public keys are generated locally

...and then sent to public key server. A buffer exists in User's installation and in the public key server that allows for the management of multiple keys.

8. Public key servers can be easily replicated.

Multiple public key servers can be deployed, and they do not need to be synchronized.

9. The public key servers do not contain sufficient information to decrypt any message protected by the system.

The public key servers do not contain any information required to decrypt any message encrypted by the system. The implication is clear: even if one is compelled, legally or otherwise, to open a message, it can be shown that it is technically impossible to do so on any server used in the system. Messages can only be decrypted at the endpoint (message destination).

10. Automatic synchronization of all User's devices.

Because of the critical need for synchronization between all of a User's devices, under the currently operative paradigm, the introduction of a new device (e.g. a new smart-phone) by a User would be problematic; a sent message that is not received on any one of a User's devices would cause the system to lose synchronization. CAPZUL's new methodology is designed to auto-synchronize all the devices of the User.

11. The methodology is independent of the cryptographic algorithms used.

Specifically, one can use any known (open or closed) asymmetric and symmetric algo-rithm. The independence of the new methodology from the cryptographic algorithms being used is critically important. This capability provides two significant benefits:

When introducing a new algorithm, especially if it is for asymmetric cryptography, it is very time consuming and difficult to prove and thus accept its efficiency and safety, making it an onerous process that has to be undertaken in order to conclude that the new method is better than the algorithm already in use. One is, however, able to conclude that the safety and efficiency of the algorithm used in the new methodology is in fact greatly enhanced in comparison with the same algorithm without the new methodology.

If a new algorithm that has been introduced does in fact show much better results than the ones in current use, it can easily be incorporated with this new methodology.

12. The public key has both fixed and variable components.

The fixed component has one or more of the key owner's identifications, e.g. an email address. This provides an external certification of the ownership of the key. The variable part changes frequently, greatly diminishing the likelihood of a successful brute force attack. Only the key in its entirety (that is, both fixed and variable components) can be used in the encryption process.

13. Exponentially increase the security level of any asymmetric encryption algorithm used in the new methodology.

Due to all the new elements introduced with the new methodology, the breaking of encrypted data is rendered futile because there is not enough information to initiate various kinds of attacks. For example, when trying to break a private key of an asymmetric encryption, one requires substantial computational power. With this new solution, one would be required to use the equivalent computational power for each newly generated public and private key. As this occurs several times per second, the multiplier-of-difficulty effect is daunting. Furthermore, in the most unlikely scenario of a successful break of the asymmetric part in the new methodology, one would still be faced with the need to break the symmetrical parts. Due to the strong synchronization between the users, where the keys are constantly being exchanged, only the communicating Users know the correct keys that were used for each message. With CAPZUL's new methodology, the inherent difficulty level of a man in the middle attack would be exponentially increased, to a level that would make such an attack impractical and ultimately, ineffective.

14. Provides the functionality of asymmetric encryption with the security level of symmetric encryption.

In order to break the security of a message, it is always necessary to break the asymmetric encryption and at least one symmetric encryption. Unlike almost all solutions on the market that use asymmetric encryption to encrypt a random symmetric key (in which case it is sufficient to break the asymmetric encryption key to obtain the symmetric), in the new methodology, the asymmetric and symmetric encryptions are not directly connected. Therefore, if one is in fact broken, the other will not be exposed. Further consideration has to be given to the use of a public key unique to each message that effectively renders the public key secret. All specific types of attack against asymmetric encryption are made impossible – the public key is not known to those attacking the encryption.

The variable part of the public key is generated by an algorithm that uses among other methods, random number generators, making the keys unpredictable, i.e. there is no pattern in the generated keys. The generation of key is based on random generators. As well, part of the data in the public key is symmetrically encrypted.

AUTOMATIC CERTIFICATION ELEMENTS

There are three levels of security for certification:

1. After the user has entered and registered in the system, and his/her identity has been confirmed by an external procedure that will have been included in the process.

2. After at least one public key has been generated for each PKS, the user will be automatically certified by the PKS and only this user's UD will be able to generate new public keys for this user.

3. After the first exchange of messages between two users, the inherent synchronization between two users is guaranteed by the use of the **synchronized shared key**, which has the internal mechanism to be changed automatically and recognized by only the two correspondents.

   For this self-certification and data encryption to be hacked, it would be necessary that all historical communication that has taken place between the two is available and is then successfully broken.

As UD uses the same cryptographic methodology to communicate with each PKS [recursion], there is extremely strong certification between the UD and all the Public Key Servers used by that UD.

The mechanism that verifies the authenticity of the UD that is generating a new public key checks if the UD is an authorized one, and also confirms the authenticity of the PKS.

After the first message exchange between two users, any attempt to introduce an unauthorized PKS or PKIS into the process [as part of an effort to hack the system] will be immediately recognized by the users.

IMPLEMENTATION

PUBLIC KEY SERVER

The PKS constantly receives the keys generated by users at their UD, and stores them in the built-in buffer for each user.

The PKS distributes these keys when requested by the UD of other users who wish to communicate with another, specified user.

The number of a user's keys buffered in the PKS will determine if, and how frequently, any one key is used more than one time.

When new keys are received from a user's UD, the PKS will certify that the origin of the keys is from the authorized source [user] of those keys.

Certification of the new key generator is made using symmetric encryption, in such a way that only the authorized UD is able to provide the correct data to PKS; PKS will use the symmetric encryption to verify if the data is correct; the UD will receive data from the PKS and will use the symmetric encryption to verify if PKS is the correct one.

All communication between a UD and PKS, other than when the UD requests the PKS's own public key, is encrypted using CAPZUL [it is a recursive solution].

The capacity of the server and the volume of keys to be served per second determine the number of users in each PKS.

The PKS address is provided to all PKIS in the system.

For redundancy and desired performance enhancement [e.g. geographic proximity], each user is able to distribute keys to various PKS; however, the keys of a user in each PKS must be different.

All User Devices of a specific user will be synchronized, as they must use the same Public Key Servers.

The PKS can be public and its data is open. If data includes encrypted data, the server will not have the keys to open this data, the only exception being the encrypted data from the server itself.

The Public Key Servers need not be synchronized with each other.

PUBLIC KEY INDEX SERVER

Each PKS sends the list of all the IDs on their server to the PKIS.

Communication between the PKIS and the PKS will only occur when there is a change, i.e. an addition or deletion of IDs in any of the PKS. This communication is secured by CAPZUL as well, as the PKS and PKIS have their own keys.

All communication between the PKIS and users is secured by CAPZUL. The key of PKIS can only be provided by the PKIS itself. For all other public key queries by users, the PKIS provides the address of all PKS used by a specified user.

All communication between a UD and PKIS is encrypted using CAPZUL [recursive solution].

When a UD sends a request for another user's key, it stores the response in its internal cache. This eliminates repetitive requests for the same user ID and optimizes performance.

As the traffic of data to and from the PKIS is minimal, the determining factor for the memory requirement is the number of users listed in the PKIS. *As an example, if there are one billion users listed in all PIKS, the memory required will be approximately 100 Giga Bytes.*

Redundancy is the only reason to have more than one PKIS; the volume of queries is so low that performance is not an issue.

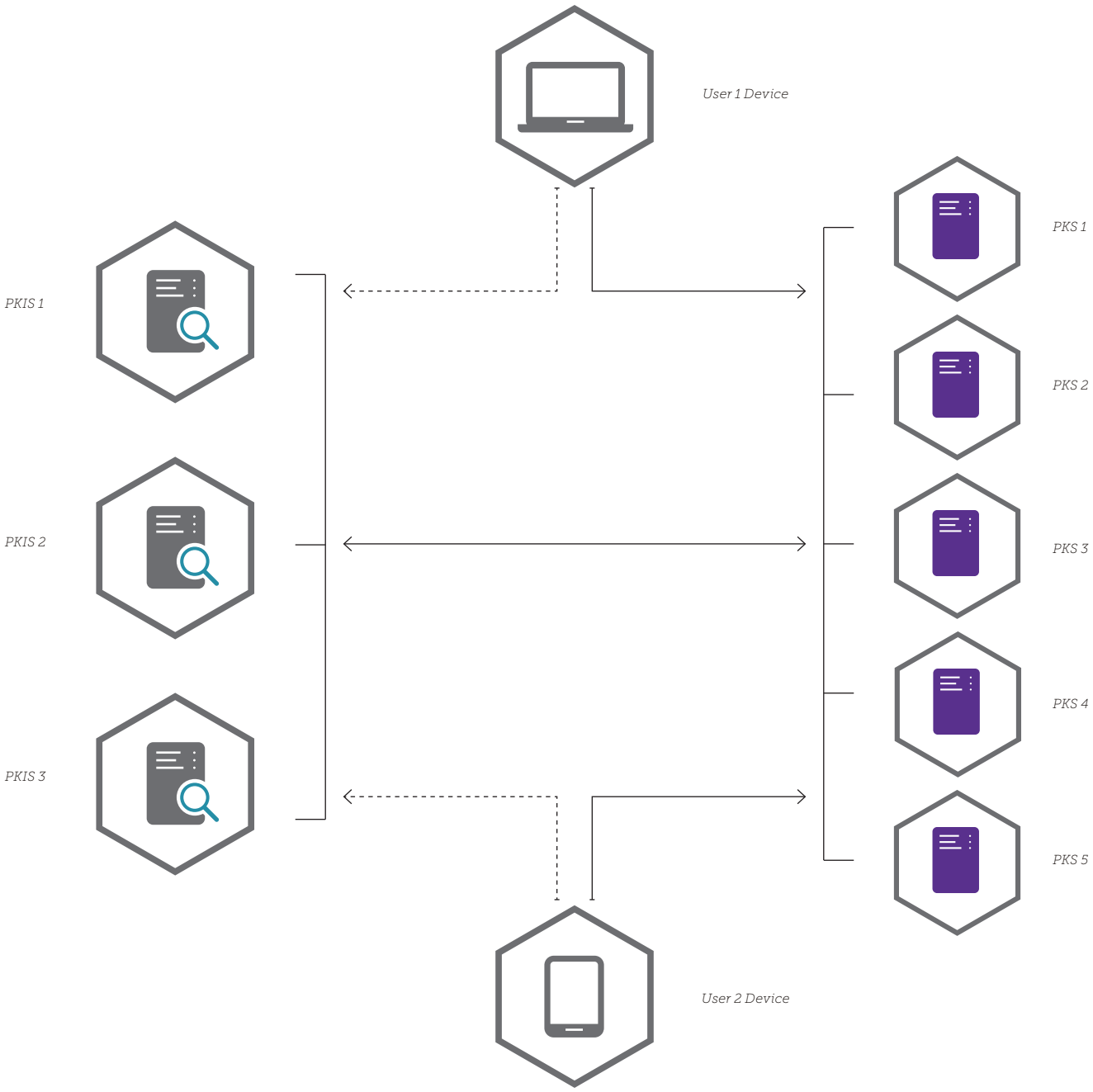
The PKIS can be public and all data stored in it is openly available. If for any reason this includes encrypted data, the server will not have the keys to open this data, with the exception of the encrypted data from the server itself.

There does not need to be a specific action to synchronize the PKI Servers with each other, as the Public Key Servers already keeps them synchronized.

The PK and the PKI Servers, and the software utilized in both, are essentially the same, with only the configuration needing to be changed.

DEVICE COMMUNICATION WITH PKS AND PKIS

Public Key Servers and Public Key Index Servers are the only external hardware required for the deployment and implementation of a CAPZUL-secured application/solution/service. The intra-system communication that CAPZUL requires occurs when a user is I) accessing the required information of a new correspondent from the PKIS and PKS, and II) when additional correspondents' public keys are required because none are left in the UD cache.



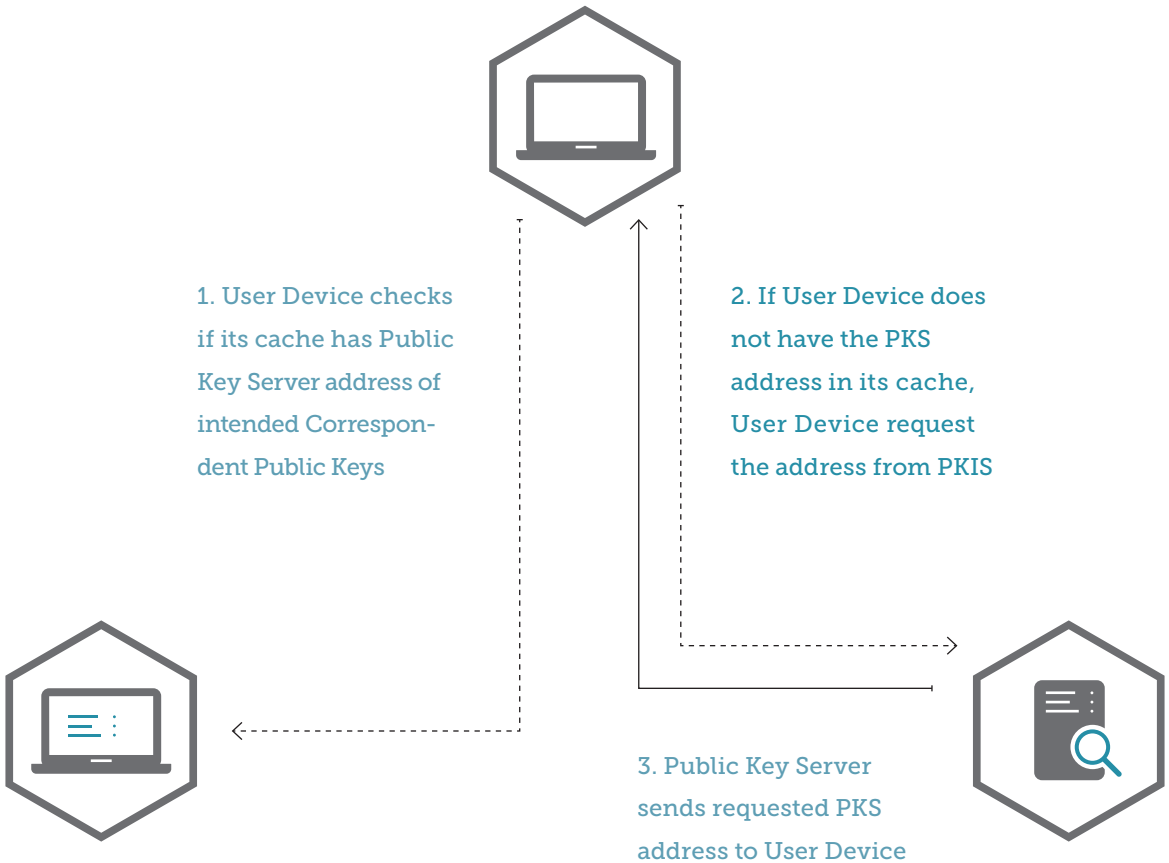
ACCESSING NEW CORRESPONDENTS' PUBLIC KEYS

When the UD needs another user's key, it first searches its own cache for the address of the PKS that stores that user's keys. If it is not found there, it will access the PKIS to get the address.

If the address of the other user is in the cache but an error is sent from the PKS, then contact will be made with the PKIS to access the new address.

The UD cache is able to store the PKS of thousands of its correspondents.

ACCESSING NEW CORRESPONDENTS' PUBLIC KEYS



ENCRYPTING & SENDING DATA

UD checks its cache for PKS address of intended correspondent;

If address of PKS exists, send a request to the PKS.

If there is not in the cache or if this request fails, send the request to the PKIS.

The PKIS returns the addresses of PKS requested for the intended correspondent.

UD stores the response [the PKS address of the intended correspondent] from the PKIS in its cache.

UD requests a key for the intended correspondent from the PKS.

PKS provides a key of the intended correspondent. When more than one key is in the buffer of the PKS, this will be a secret [public] key, known only to the sender and intended correspondent.

UD generates a **temporary symmetric key** – to be used for one transmission only.

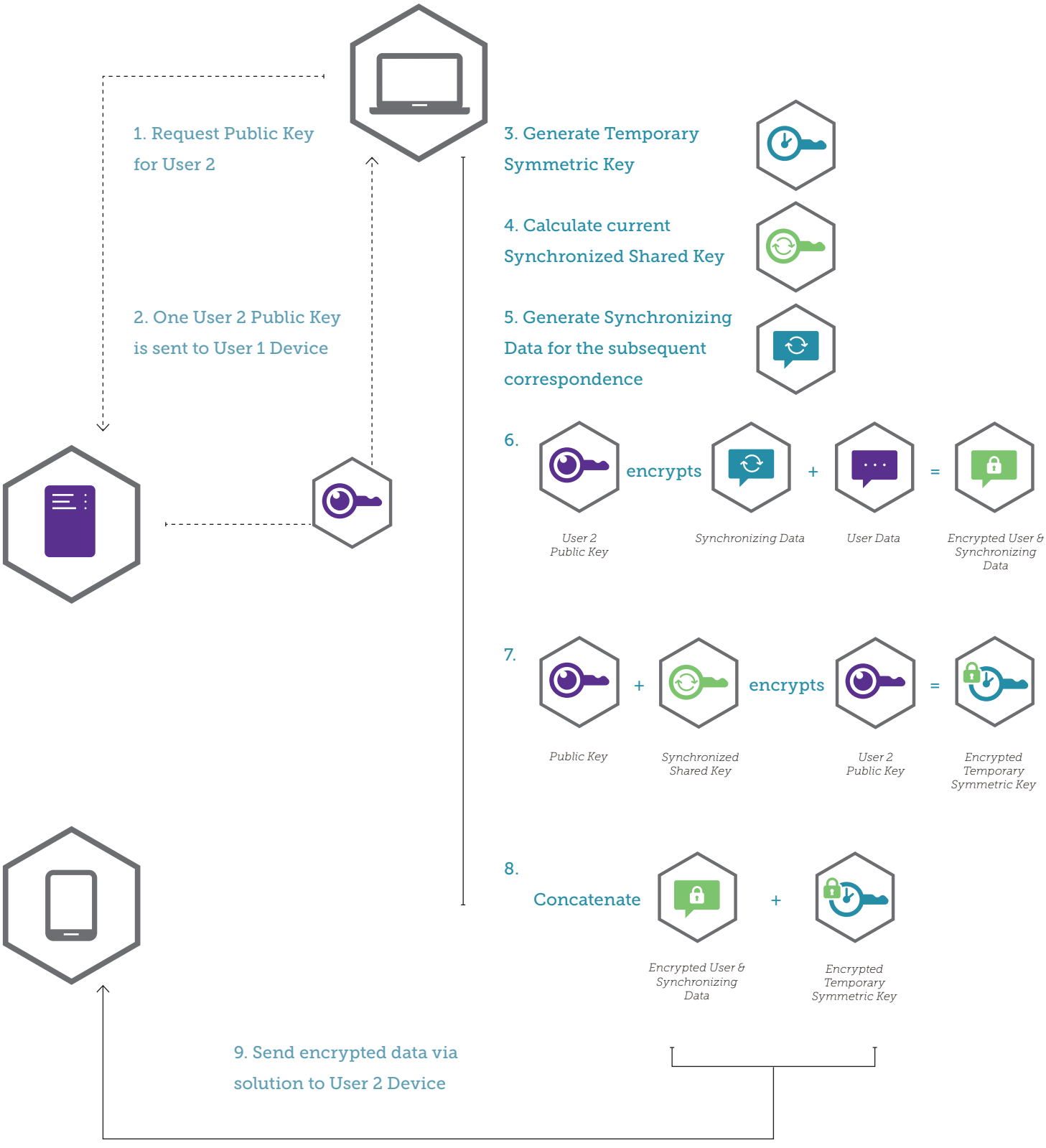
UD calculate a **synchronized shared key** and generates new synchronizing data in order to synchronize the next communication between these users.

UD uses the **temporary symmetric key** to encrypt the data [the original message] + synchronizing data.

UD uses the public key + the current **synchronized shared key** [between sender and recipient] for encrypting the **temporary symmetric key.**

The encrypted **temporary symmetric key** + encrypted data [message data + synchronizing data] are concatenated and sent by the regular channel through which the original message would be sent without encryption, e.g. Gmail.

ENCRYPTING & SENDING DATA



RECEIVING & DECRYPTING DATA

UD receives encrypted data through any normal channel, e.g. Gmail.

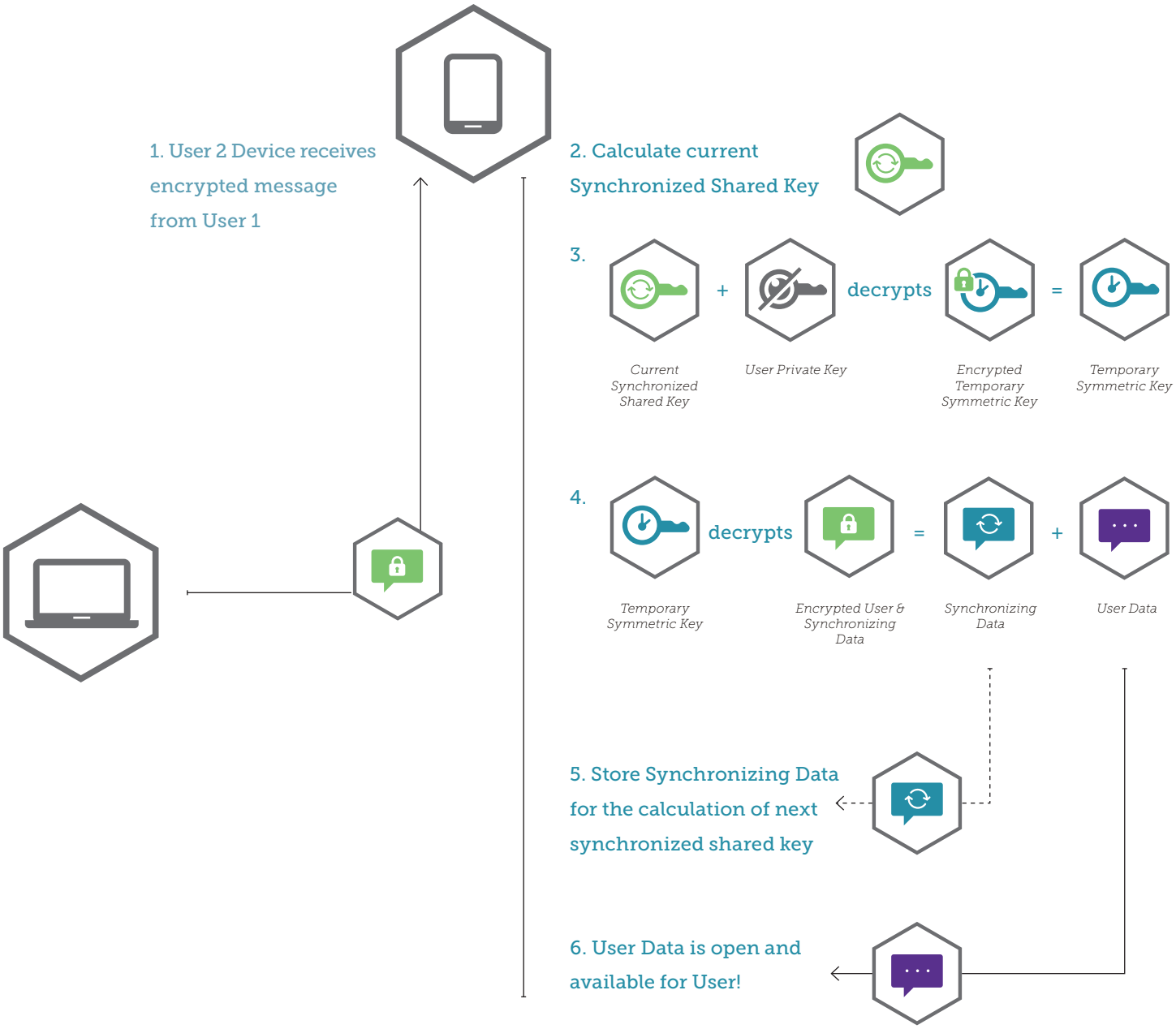
UD calculates the *current* **synchronized shared key.**

The *current* **synchronized shared key** + **private key** [associated with the public key that was used to encrypt the data] is used to decrypt the encrypted **temporary symmetric key.**

The **temporary symmetric key** decrypts the encrypted data.

UD stores the synchronizing data for calculation of the *next* **synchronized shared key** that will be used to communicate the next messages with this same user.

RECEIVING & DECRYPTING DATA



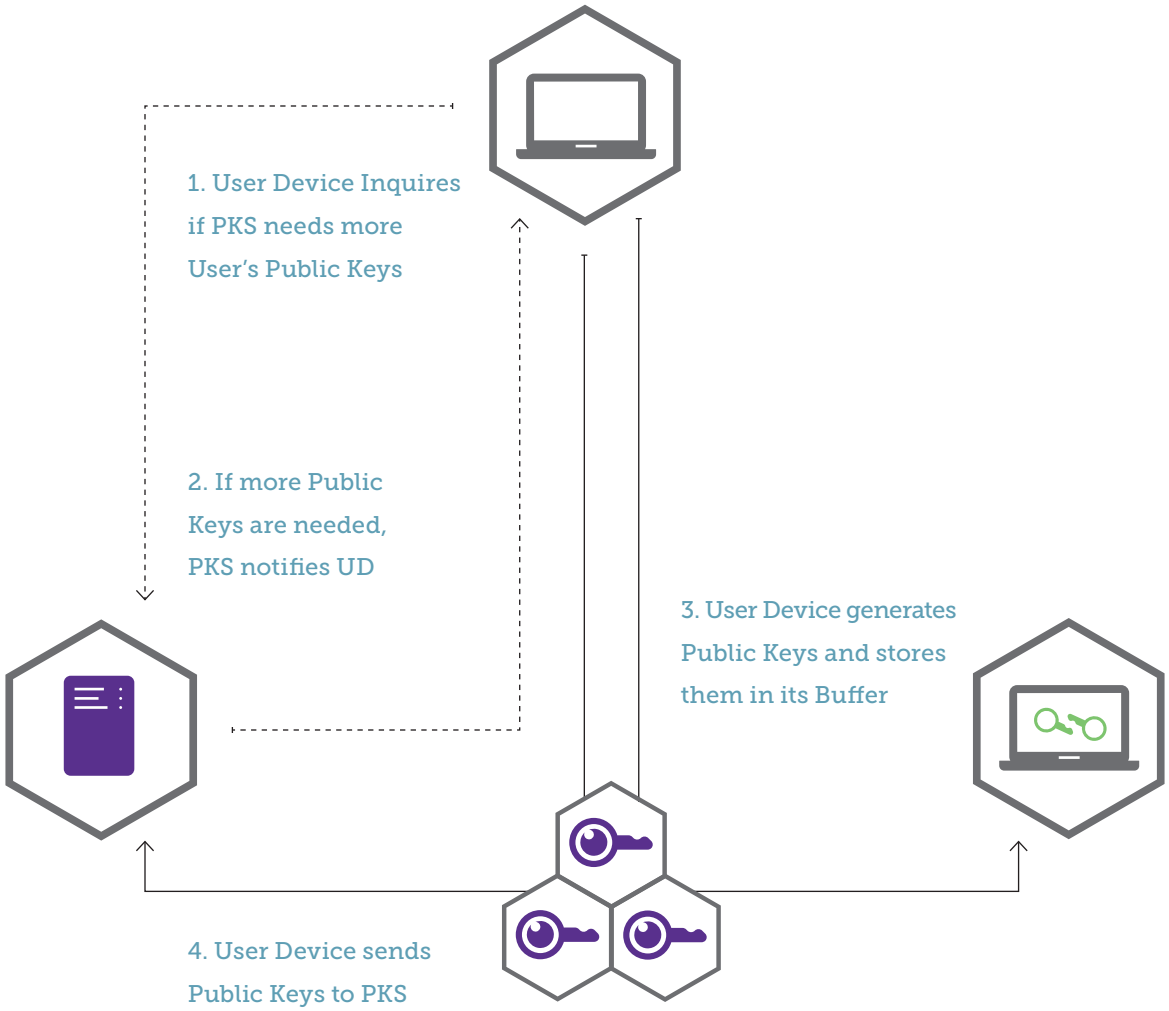
GENERATING & SENDING NEW PUBLIC KEYS TO PKS

From time to time, UD communicates with PKS to find out if PKS needs new public keys.

If there is no communication between UD and PKS, the UD attempts to connect to the PKS from time to time, until successful.

When necessary, UD generates new public keys at such time as it will not degrade the device's performance, and then sends them to the PKS.

GENERATING & SENDING NEW PUBLIC KEYS TO PKS



ICON GLOSSARY

User Private Key

Asymmetric cryptography key, generated at the User Device, only known to the user

User Public Key

Asymmetric cryptography key, generated at the User Device, known only to the correspondents

Temporary Symmetric Key

Symmetric cryptography key, generated at sender's User Device, used to encrypt User Data & Synchronizing Data at the sender's device

Synchronized Shared Key

A symmetric key, calculated at a User Device, using historical synchronization data between correspondents

Encrypted Temporary Symmetric Key

The Temporary Symmetric Key that is encrypted by the recipient's Public Key + the Current Synchronized Shared Key - this Synchronized Shared Key will only be used for this one transmission

Public Key Server

Public Key Server that has a cache of users' multiple public keys

Public Key Index Server

Public Key Index Server lists all CAPZUL User IDs received from every Public Key Servers

User Data

Message data [text/audio/video] to be transmitted

Synchronizing Data

Data generated in the User Device in order to synchronize the next communication between Users 1 & 2

Encrypted User & Synchronizing Data

The User Data + Synchronizing Data that is encrypted by the Temporary Symmetric Key

User 1 Device

User Device transmitting CAPZUL-encrypted data

User 2 Device

User Device receiving CAPZUL-encrypted data

User Device Cache

Cache in the User Device that stores the Public Key Server addresses of its correspondents

Public Key Buffer

Buffer in the User Device that stores its own Public Keys that it generates, to be then sent to PKS