cameoNet Security Whitepaper

Michael Merz (dermicha@cameo.io)
Björn Reimer (reimerei@cameo.io)

28.11.2014, DRAFT Version 0.1

Contents

Overview

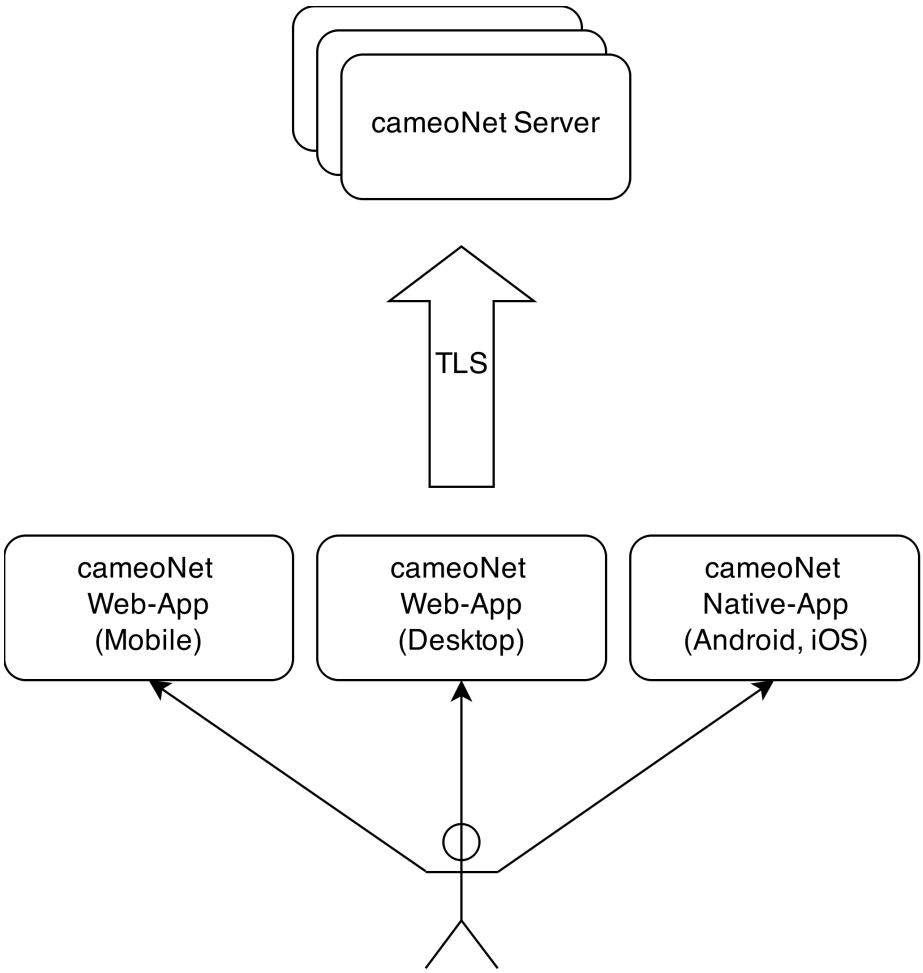

Figure 1: cameoNet Overview

cameoNet ist ein Multi-Plattform/Device/Identitäten Messenger. Mit cameoNet können Nutzer auf einem sehr hohen Sicherheitsniveau gegen Datenmißbrauch und ungewolltem Mitlesen von Dritten schützen. cameoNet ist offen für die leichte Einbindung von externen Kontakten (per Mail oder SMS). Der cameoNet Quelltext ist vollständig veröffentlicht.

Implementation Details

Source Code

Der vollständige Source Code steht bei github zur Verfügung: https://github.com/memoConnect

Der Source Code des cameoNet Server ist in folgendem Repository zu finden: https://github.com/memoConnect/cameoServer

Der Source Code des cameoNet Client ist in folgendem Repository zu finden: https://github.com/memoConnect/cameoJSClient

Encryption Frameworks

Die gesamte Verschlüsselung erflolg auf der Client-Seite. Dazu werden foldende externe Framework eingesetzt:

- JSEncrypt, OpenSSL RSA Encryption, Decryption, and Key Generation
- CryptoJS, SHA256
- Stanford Javascript Crypto Library (SJCL), AES
- [OpenSSL (iOS App)] (), OpenSSL RSA Encryption, Decryption

Transport Encryption

Ergänzend zu der Verschlüsselung aller Nutzerdaten wird der Transportweg zwischen Client und Webserver TLS basiert verschlüsselt. Das eingesetze SSL Zertifikat ist ein "extended Validated" Zertifikst der CA COMODO. Für die Signatur des Schlüssels wurde SHA256 verwendet.

Die TLS Konfiguration der Webserver wurde auf hohe Sicherheit und möglichst breite Browser Unterstüzung ausgerichtet. Eine Validierung wurde mit Hilfe der frei Verfügbsren Tools von SSLLabs und SSLZilla durchgeführt.

Zusätzlich werden denkbare Attacken auf TLS Verbidnungen durch den Einsatz von DNSsec und DANE weiter erschwert.

Security disclaimer: it is planned to add certification pinning to cameoNet clients.

Account/Identity Modell

Account

Each user has at least one account. cameoNet does nothing to avoid that users create more than one account. An account consists at least of a username and a

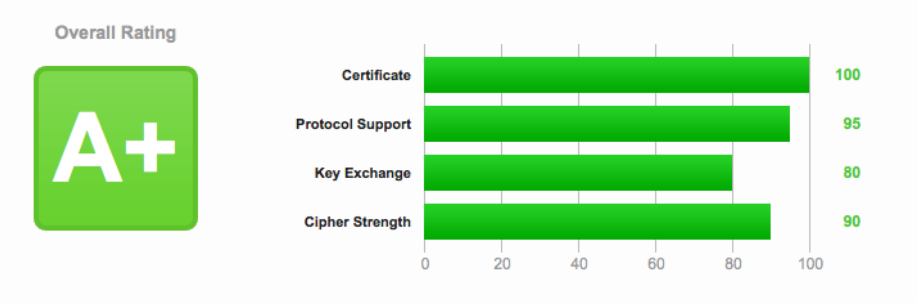

Figure 2: SSLLabs TLS check result, 28.11.2014

password. The username must have at least 3 characters. A user hast at least one Identity per account. The first identity will be created during registration.

Identity

A user get's in touch with other users only through one of his identites. Each Identity has its own contacts/keys/talk. Each identity of a user is complete indepentent. Other users could not determine which identites belongs to which account.

Data Encryption

cameoNet uses AES and RSA encrption. AES will be used with a key length of 256 bit. RSA keys have at least a length of 2048 bit.

All messages and assets will be encrypted using AES. A new random AES key will be created for each talk. All messages and assets of a talk will be encrypted with the same talk based AES key.

The AES key will be encrypted with all public keys of the recipients. In the case Alice uses a smartphone and a PC and Bob uses a PC and a tablet the AES key will be encrpted 4 times. That makes it possible for Alice and Bob to read this talk on all their devices.

Encryption Levels

In the case a user sends a message to an external contact cameoNet provides two altertnatives to encrypt the content. In both cases all content will be enrypted using AES256. The two alternatives differ only in the was the AES key will be exchanged. Both is usable without to enforce the external contact to register at cameoNet

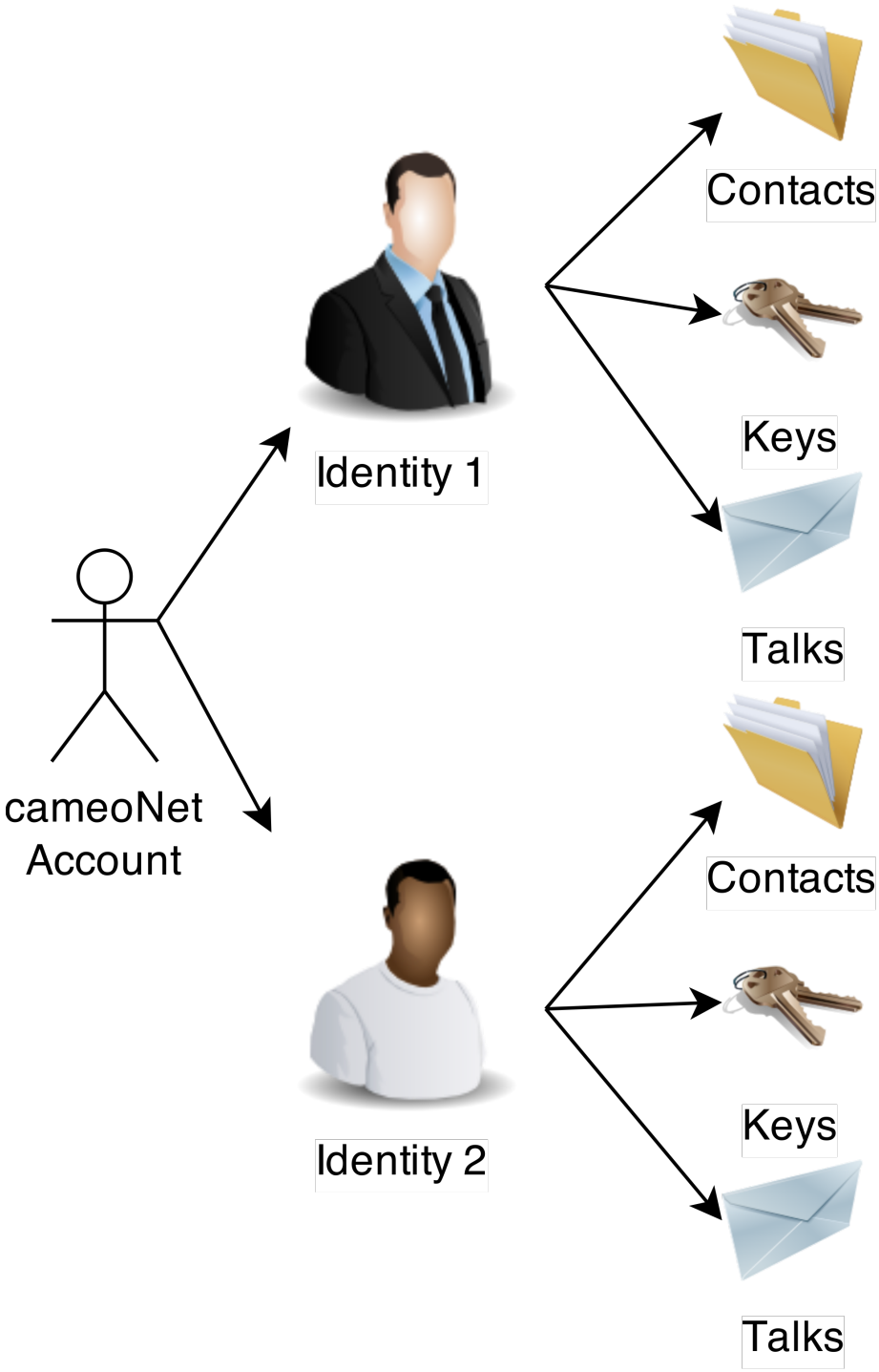

Figure 3: cameoNet Identitites

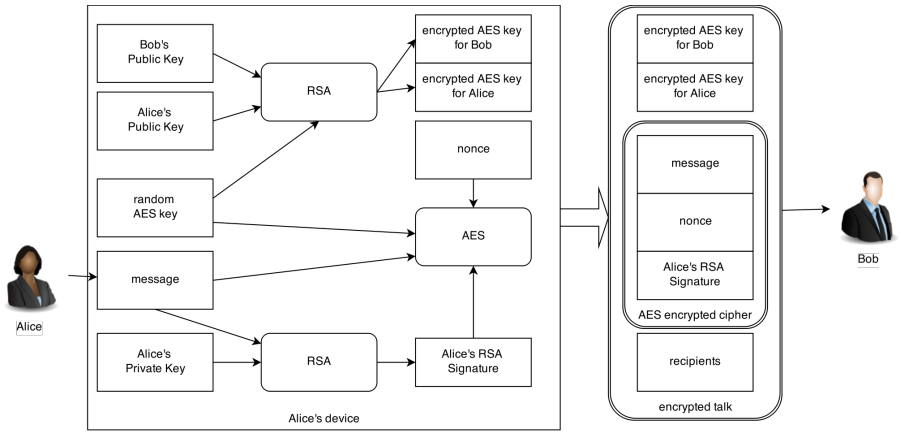

Figure 4: cameoNet Crypto System

manual AES Key Exchange

In this case a cameoNet user could define an AES Key. He has to transfer this key secure to all external contacts.

External contacts will get an email oder SMS which includes a link (personal URL => PURL). After opening the PURL external contacts have to enter the AES key. That enables external contacts to read and write messages.

Security disclaimer: it is planned to pin a PURL to the first browser which opens the PURL.

PassCaptcha based AES Key Exchange

In this case a camepNet user cloud define an AES Key and a captcha (cameoNet PassCaptcha) which contains this key. This captcha will get part of the talk.

External contacts will get an email oder SMS which includes a link (personal URL => PURL). After opening the PURL external contacts have to enter the AES key, which will shown as a PassCaptcha. That enables external contacts to read and write messages.

Security disclaimer: it is planned to pin a PURL to the first browser which opens the PURL.

Key Pair Verfification

Key authentication is used to establish trust between two keys. This is done between multiple keys of one identity.

The following is assumed before an authentication is started:

- The public keys have been exchanged via an insecure channel
- The cameoId of the owner of each key is known

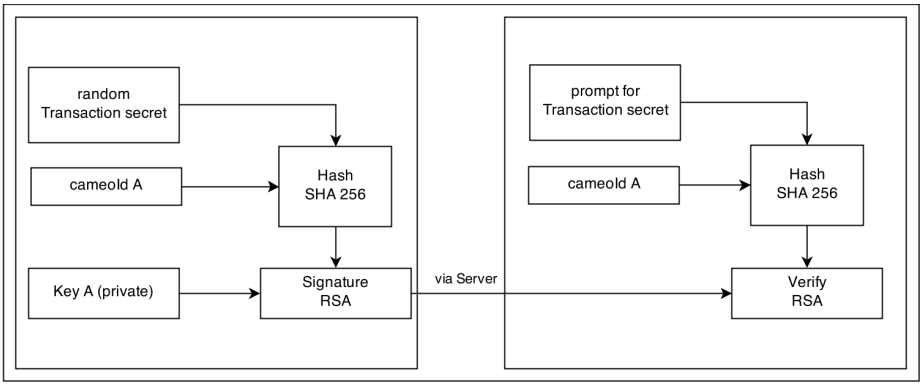

Figure 5: Handshake

When the authentication was successful the authenticated key will be signed. Future conversations with this key will be marked as trusted.

After the user created more than one RSA key pair he has to verify them. This could be done by having two devices active, one with the new key pair and one with an already verified key pair. The verification wil be done by manually transfer an 8 digit id. It is only nessesary to verify a new key pair with one already verified key pair.

To verificate a key pair means to sign its public key.

Verfification between Identites

User can initiate verification of an other cameoNet idenitites. The process is the same than veryfing hoes own key pairs. It is also only nessesary to do verification between one key pair of each involved identity.

After two indentities have verified their keys they can communication with the highest level of security.

Random Numbers

Zufallszahlen werden im Client erzeugt. Dieses erfolgt immer mit der besten zur Verfügung stehenden Methode. Bei der Auswahl des Zufallszahlengenerators wirde diese Reihenfolge eingehalten:

1. window.crypto.getRandomValue: vom Browser zur Verfügung gestellter Zufallszahlengenerator. Implementation abhängig von Browser und Betriebssystem. Aktuell wird die Qualität als ausreichend eingeschätzt.
2. Zufallszahlengenerator der sjcl. Die Entropie wird ab Start der App aus Nutzereingaben, DOM-Elementen und Uhrzeit gesammelt.
    1. Wenn nicht genug Entropie vorhanden ist (in der Realität eher selten), wird Math.random verwendet. Die Qualität ist den seltenden Fällen schlecht.

Key Storage

All private keys will be stored on the device on which they have been created. As storage the HTML5 Local Storage will be used. Private keys will be stored AES encrypted. Each user has a server side stored secret, which is used as AES key to enrypt the private key.

Security Aspects

@TODO