Keeping Connected When the Mobile Social Network Goes Offline

Øystein Sigholt*, Besmir Tola†, and Yuming Jiang†
Department of Information Security and Communication Technology
NTNU-Norwegian University of Science and Technology
Trondheim, Norway
Email: oysteils@stud.ntnu.no*, {besmir.tola,yuming.jiang}@ntnu.no†

***Abstract*—WiFi Direct is an embedded technology in a vast majority of smartphone devices running the Android operating system. As a result, it represents a promising technology that can be exploited in re-establishing connectivity among user devices in case of cellular network outages. A technique that smart devices can use to restore connectivity in situations where they are unable to connect to a cellular tower or access point, but close enough to support device-to-device communication is presented. The proposed technique envisions a combination of security layers that ensure authentication, confidentiality, and integrity of communications among end users. Each device is issued a certificate by a central authentication entity at sign up and when it is unable to connect to the server component, it will attempt to form a group with nearby devices in the same situation over WiFi Direct. Once a WiFi Direct group has been formed, the group owner will temporarily assume the role of the server, and each group member and the group owner will verify each others identity and connect using mutual Transport Layer Security (mTLS), facilitating secure communication. The approach is validated through the implementation of a mobile social application involving several mobile devices, and overheads due to the additional security features are investigated.**

***Index Terms*—Peer-to-peer Communications, WiFi Direct, mTLS, Mobile Social Networks**

I. Introduction

Broadband cellular networks are becoming the most dominant means for mobile data access world-wide. According to a recent mobility report by Ericsson [1], mobile data traffic is expected to undergo an annual growth of around 31% over the coming years. Among the top mobile application categories, video traffic followed by social network applications cover almost 75% of the total monthly data traffic. However, when a mobile user moves out of cellular coverage, or is unable to reach the central server of a service for any reason, connectivity is lost. As a consequence, Internet-based mobile applications will not be able to provide their services and the end-users will become isolated until Internet connectivity is restored, even if the users that are making use of the service, are within a few meters of each other.

Modern smartphones are equipped with a number of radio interfaces that enable wireless communication among devices in close proximity. These capabilities can be used to establish connectivity between neighboring devices even in the most remote out-of-coverage locations and in cases where cellular network outages are experienced.

WiFi Direct [2], a wireless technology allowing WiFi devices to connect directly to each other in a Peer-to-Peer (P2P) fashion, is one of the ways that nearby mobile devices can establish connectivity. It is particularly widely available due to it not requiring any specialized hardware apart from a typical WiFi radio, and easy to use as addressing can be done using the familiar Internet Protocol. In order to establish communication, a common setup involves the formation of a WiFi Direct group of peers where one of the peers acts as a software Access Point (AP) to the remaining devices. This device is referred to as the Group Owner (GO) and the other devices associated to the GO represent the Group Members (GMs). The group formation can be achieved in three different procedures denoted as *standard*, *autonomous*, and *persistent* group formation. During each of these procedures a number of actions are taken by the WiFi Direct capable devices for performing device discovery, GO negotiation, service discovery, security provisioning and address configuration [3].

WiFi Direct connections entail WiFi Protected Setup (WPS) as means to provide a secure connection among members through some manual intervention like inserting a PIN or PushButton Configuration (PBC). This way, users are able to authenticate themselves in the network. However, there are services that require stronger levels of security where data confidentiality, integrity and authenticity are of utmost importance. Mobile Social Networks (MSN) providing Instant Messaging (IM) services necessitate enterprise and automated authentication methods such as Extensible Authentication Protocol - Transport Layer Security (EAP-TLS).

Even though WiFi Direct can be used to reestablish connectivity, a social application usually depends on a central server that users trust. When connected to this server, users have confidence that their messages are delivered to their respective recipients and that no other users of the service can access their private conversations. Without access to this central server, a user in an out-of-coverage P2P context must verify the authenticity of their peers themselves.

The scope of this work is therefore to propose, implement, and experimentally validate a combination of upper layer measures that can be used to securely and easily enable authenticated communication in existing social applications, also in situations with no Internet connectivity.

The remainder of this paper is the following. Section II

presents the related work. In Section III, we illustrate the proposed system architecture, and the relative components for enabling secure and trustworthy communication over WiFi Direct. An implementation on a real testbed, composed of several smart devices running the Android operating system (OS), and how the system components interact with one another are presented in Section IV. Successively, the validation and experimental results analysis of the different security layers adopted in the architecture are illustrated in Section V. Discussion regarding overheads, incurred due to the additional security levels, in terms of both computation and connection times are presented in Section VI. The potential and limitations of the proposed architecture in regard to connectivity, security, user experience, and overhead are discussed in Section VII. Finally, Section VIII concludes the paper.

II. Related Work

A number of commercial applications for P2P communication, applying various combinations of WiFi and Bluetooth, exist for the Android ecosystem. The most prominent, FireChat, made headlines in 2014 when it accomplished half a million downloads over a period of two weeks as Hong Kong protestors used its P2P functionality to organize efficiently even in areas with heavily congested network traffic [4]. An open source privacy-focused, decentralized, alternative to FireChat, Briar, also uses Bluetooth and WiFi to communicate in addition to the Tor network. It uses public key cryptography to manage identities and secure the communications link, but its decentralized nature and lack of a universally trusted entity makes exchanging identities a difficult problem that in practice requires the two parties to physically meet and manually exchange keys before a connection can be made [5].

A large-scale research effort by the name of the Serval Mesh aims to create an independent network by relying on WiFi devices to form a mesh network based on WiFi ad-hoc mode [6]. Unfortunately, WiFi ad-hoc mode is mostly unavailable on consumer smartphones without modifications, thus making it unsuitable for many use cases.

Shahin and Younis present a framework for P2P networking of Android devices using WiFi Direct that covers discovery, connection establishment, peer management and communication between peers in a single group [7].

Wong et al. noted that connecting Android devices with WiFi Direct uses WiFi Protected Setup (WPS) which requires manual user interaction to accept the connection prompt. They propose using WiFi Direct to create APs that advertise their Service-Set Identifier (SSID) and Pre-Shared Key (PSK) using Network Service Discovery (NSD) instead of leveraging fully fledged WiFi Direct connection establishment. Any WiFi capable client can then connect to the WiFi Direct AP like they would connect to any other WiFi AP (referenced as connecting as a legacy client in WiFi Direct terminology) without the need for users manual verification [8]. They do not however, consider the authentication aspect covered by this work.

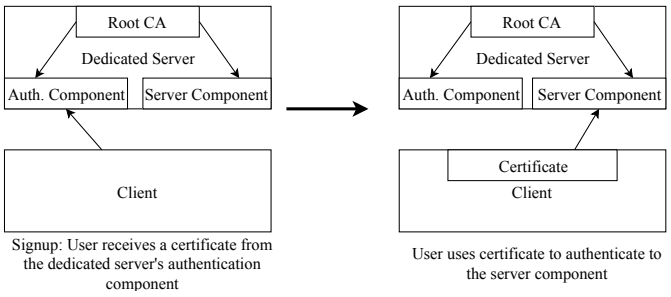

Fig. 1. A client signing up to the service by obtaining a digital certificate from the authentication component and using it to authenticate to the server component.

III. System Architecture

A system consisting of three logical components is proposed. These components can be implemented to facilitate secure communication between users both when connected to the Internet and when out-of-coverage.

A. The Authentication Component

The authentication component is the single mutually Trusted Third Party (TTP) among the members of the social network with the sole responsibility of managing the identities of the users. This component is only available over the Internet, and can therefore not be reached in out-of-coverage operation.

Figure 1 illustrates the signing up to the service process which is done in the same fashion as in a traditional Public Key Infrastructure (PKI) system. The client generates a key pair and creates a Certificate Signing Request (CSR) that is sent to the authentication component for signing. The CSR contains the public key as well as the identity (or Distinguished Name (DN)) that the certificate is for.

The DN should contain some human readable component that users can later use to distinguish between their contacts. This can for example be a username or an email address.

After the authentication component has approved the CSR, the applicant is issued a signed X.509 digital identification certificate that contains the DN, the users public key and a signature that binds the public key to the DN. This means that the authentication component has approved the public key contained by the certificate as belonging to the specific DN also contained in the certificate. The resulting certificate can be used to authenticate to other entities in the system, offline or not as shown in Figure 1. To revoke credentials after the fact, a Certificate Revocation List (CRL) can be maintained (see [9, Section 3.3]).

B. The Server Component

The server component is responsible for message forwarding to connected clients both over the Internet and during out-of-coverage operation, i.e., WiFi Direct operation mode, and accepts incoming TLS connections on a predefined port.

By exchanging certificates and verifying that they are issued by the authentication component, both the client and the server confirm that the other is a registered user of the service, and

learn the other party's identity. The server component then maintains a forwarding table linking the connected clients identity (public key) to the appropriate socket.

1) Out-of-coverage Operation: The server component can either run on the dedicated server (introduced in more detail in Section IV-A) and be available over the Internet, or on a user's device in an out-of-coverage situation. When running on a GO it is also the server component's responsibility to set up and manage P2P connectivity to nearby devices. Figure 2 shows how two devices form a P2P group and reestablish connectivity if the dedicated server is unreachable.

If the user device is unable to locate any other nearby P2P devices hosting an instance of the server component, the server is instantiated on the device in out-of-coverage mode. It will autonomously form a P2P device group (middle part of Figure 2), and broadcast the information needed to connect to said group using Wong et al.'s technique [8].

When the group is formed (bottom part of Figure 2), the device will accept incoming connections from nearby devices and the server component will manage message routing just like it would during regular operation on the dedicated server.

If Internet connectivity were to be restored at any time, a device will simply detect the connectivity change and reconnect to the dedicated server, tearing down any open P2P connections.

2) Message Routing: A message packet with the senders public key, the recipients public key, a timestamp, a ciphertext and a signature is transmitted by the clients to the server when they wish to communicate with another user. The recipients public key indicates which client the message should be forwarded to and the signature (protecting the integrity of the other fields) is verified using the attached public key belonging to the sender.

Caching and retransmissions of messages are the responsibility of the client, making the operation of the server rather simple. If signature verification fails, the message is discarded. If not, the server examines its active connections and checks if a client has connected with the identity of the receiver. Successively, the message is forwarded to the appropriate client. If the server does not have an open connection to the correct recipient, the packet is also discarded. Relying on servers to cache messages would cause messages to never make it to their intended recipients if a GO were to disconnect before being able to forward it.

C. The Client Component

The client component is responsible for managing messaging and connecting to and authenticating the server component.

1) Connectivity: The client component is expected to connect to and disconnect from multiple server component instances in a single session as a device might move in and out of range of cellular coverage and P2P groups.

If not connected to a server component the client will first attempt to establish a connection to the dedicated server via the Internet. At the same time, it will start the device discovery process to locate nearby P2P groups. If the dedicated server is not reachable and no group is found, it will set up an instance of the server component and form its own P2P group.

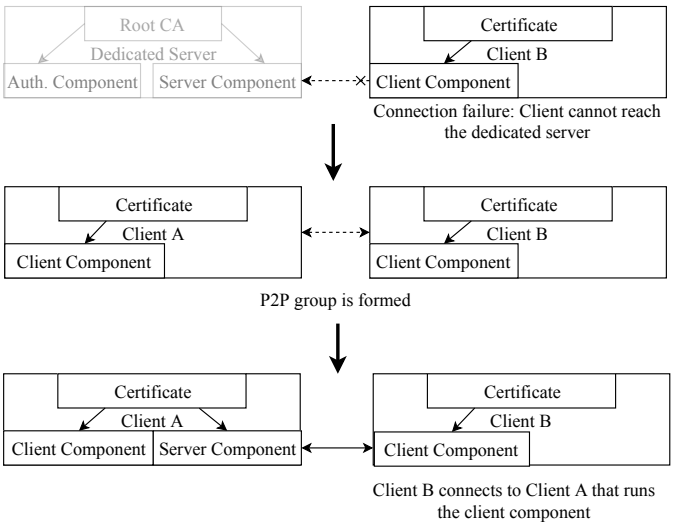

Fig. 2. A client unable to reach the dedicated server establishes a connection with a nearby device. Arrows from the certificates to the client and server indicate authentication during connection.

The first device to form a group will advertise its connection information so that nearby devices can discover and join the P2P group. Upon joining a group, the client will attempt to connect to the server component instance running on the GO, instead of the dedicated server.

The client enforces secure connections and will only connect to server components that offer identification certificates issued by the the authentication component over TLS.

2) Messaging: Message packets are transmitted to any server component as soon as possible. If the recipient does not acknowledge the message, the client assumes it was not delivered and caches it for retransmission. If the client connects to another server component it will immediately attempt to transmit all queued messages, otherwise it will periodically attempt to retransmit them with an exponential backoff strategy where the client periodically retransmits the message with increasing delays between attempts.

IV. Implementation

A. The Dedicated Server

The dedicated server is a traditional server reachable using the Internet. It implements both the server component to facilitate messaging and the authentication component to facilitate sign-up.

B. The Client Application

The client application not only implements the client component, but also the server component. It consists of two primary activities and one debug activity to inspect messages being sent and received.

1) The Main Activity: The main activity contains a list of a user's contacts and the current connection status. The connection status has four distinct values representing different stages of the application connection phase.

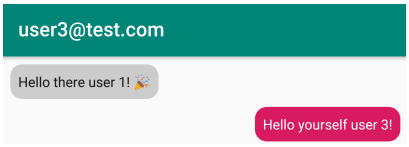

Fig. 3. Two users chatting using the **Chat Activity** as seen by `user1@test.com`.

- **Setting up:** The application is loading
- **Connecting:** Looking for a server to connect to
- **Connected:** Successfully connected to a server
- **Hosting:** Unable to connect, the server component has been started and is hosting a group

Selecting one of the contacts in the list opens up the chat activity for that particular user. Pressing the info circle in the top right of the screen opens the debug activity.

2) The Chat Activity: The chat activity (seen in Figure 3) allow users to send basic text messages to their contacts. It shows messages sent/received in chronological order. Messages are only shown if the attached signature can be verified.

3) The Debug Activity: The debug activity (seen in Figure 4) show the messages being sent from/to, or being relayed by a user. It loads the name of the users from the contact list if an entry is found corresponding to the public keys contained in the message.

It displays the state of the attached signature by verifying it using the attached sender public key, and displays a checkmark if the signature can be verified, and a cross if the verification fails. It also decrypts the message ciphertext if it possesses the private key corresponding to the recipient.

C. Cryptography

Elliptic Curve Cryptography (ECC) was selected based on the fact that it requires relatively short keys to provide strong security. It is based on the premise that it is difficult to find the discrete logarithm of an elliptic curve point [10].

Elliptic Curve encryption was done using Elliptic Curve Integrated Encryption Scheme (ECIES). To encrypt a message, a key agreement function is used to create a shared secret value based on a randomly generated ephemeral key pair and the recipients public key. A key derivation function is then used to generate symmetric keys for encrypting and signing the message contents from the shared secret. The actual message encryption is performed using a symmetric encryption algorithm and a digest function is used with the signing key to generate a Message Authentication Code (MAC). The recipient can then obtain symmetric keys to verify the MAC and decrypt the message by using the ephemeral public key and recipient private key, along with the encrypted message [11].

The MAC from ECIES only preserves the integrity of the message itself, so the Elliptic Curve Digital Signature Algorithm (ECDSA) was used to sign the entire message frame including the sender and recipient fields. An ECDSA signature is the elliptic curve variant of the Digital Signature Algorithm (DSA). It is calculated using the senders private key and a random value, and can be verified by knowing the random value and the senders public key [12].

V. Security Validation

The following section covers an analysis of what can be learned about the operation of the sample implementation and its users by examining select network layers.

A. Data Link Layer Security

As messages are the only data frames transmitted in the WiFi Direct group, it is trivial for an unauthenticated third party monitoring the network to see which devices are transmitting messages. If the aforementioned third party is able to link the physical addresses of each device to an identity, it can keep track of when messages are sent and to/from whom.

Using Wong et al.'s technique of broadcasting the PSK of the formed network using NSD to allow devices to connect to the WiFi network without the need for manual verification prompts makes connection establishment considerably easier for the user [8]. However, this also means that anyone can use service discovery to learn the PSK. A user must therefore not be considered trustworthy based only on their ability to connect to the WiFi network.

Encrypting the PSK with a static secret as done in the original paper would only make the key marginally more difficult to recover, as the secret would need to be distributed onto every users device, from which it could be recovered.

WiFi Direct connections between devices must therefore be considered insecure channels, as an adversary must be considered able to decode transmitted frames [13].

B. Transport Layer Security

As the data link layer security is not sufficient to protect data over the air, the system relies on upper layer security. On the transport layer, the data is transferred over TLS. A number of attacks on TLS have been published that can defeat this protection [14]. It is therefore vital that both the server component and the connecting clients enforce the most recent best practices for TLS [15]. With full control over the implementation of both servers and clients, strict requirements can be enforced without worrying about compatibility.

In addition to confidentiality and integrity protection, the transport layer also provides the access control that is lacking on the lower layers. As both connecting parties require authentication of the other party using mTLS it is impossible for a user to connect without an identity from the authentication component. If attempting to connect without the appropriate credentials, the TLS handshake will fail, and the device will be unable to communicate with other users.

It can therefore be concluded that unauthorized clients are prevented from communicating with users at the transport layer, and that the data being carried is both confidential and integrity protected due to the properties of TLS. It is however important to note that the TLS connections are not End-to-End (E2E), as they are terminated at the server component running on the GO. Upper layer measures are therefore required to protect users from a dishonest GO.

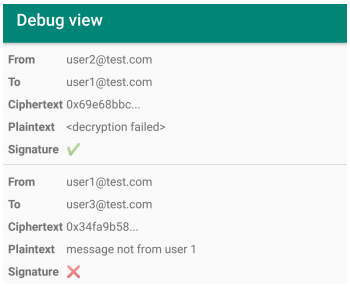

Fig. 4. Message flow as seen by the GO `user3@test.com`.

C. Application Layer Security

As seen in Figure 4 (upper part), senders and recipients are visible to the GO, and signatures may be verified, but message contents cannot be deciphered without the appropriate private keys, so the GO is unable to learn message contents (*Plaintext* field).

A dishonest server component could potentially attempt to modify the messages it is forwarding. At the bottom of Figure 4, a malicious GO has modified the *from* field of a message packet, but was unable to correctly sign the message as it does not possess the private key of `user1@test.com`. The recipient (`user3@test.com`) therefore discards the message due to the signature verification failure.

In short, the message packet format successfully protects the messages confidentiality and integrity E2E, but does not protect the identity of the sender and recipient.

The GO can, however, choose not to deliver, delay or even deliver messages multiple times. The latter, known as a *replay attack* has an especially undesirable impact if used maliciously. In a high security context, messages that are received long after they were created should be discarded or the user should be notified as a stale message could indicate malicious activity. Duplicate messages should be discarded.

VI. Overhead

The number of steps taken to secure reliable instant messaging in this research add some overhead that must be considered in the evaluation of the proposal.

A. Connection

In the event that the Internet connection fails, some amount of time is required to reform connectivity using WiFi Direct. Camps-Mur et al. [3] measured the group formation delay and noted that the WiFi Direct discovery mechanism introduces some randomness to the time it takes to connect to a group.

In our setup, timing of this delay was obtained by logging the time it took one Nexus 6P running the sample implementation to connect to another which had already autonomously formed a group and started broadcasting connection credentials using NSD. The process was repeated 500 times by a Bash script which power cycled each device, opened the application on one device, and waited for it to become GO before launching the application on the other device.

Figure 5 presents the CDF of the measurements. After a group has been formed, it takes around five seconds (discovery

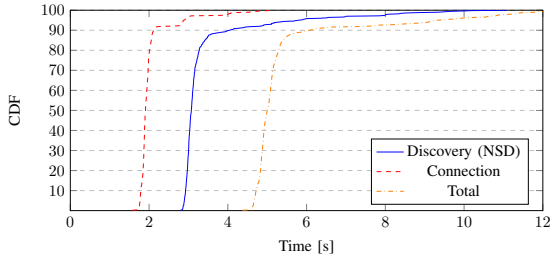

Fig. 5. CDF of the time to discover credentials over NSD and connect to a WiFi Direct group as a legacy client.

	Mean μ	Median	Standard Deviation σ
Discovery (NSD)	3.42	3.07	1.17
Connection	2.03	1.90	0.46
Total	5.45	5.00	1.54

TABLE I
Seconds used to discover credentials over NSD and connect to a WiFi Direct group as a legacy client.

delay plus connection delay) for the first GM to discover the broadcasted credentials and join as seen in Table I. This new client goes through the discovery process which can be expected to take three to four seconds, and uses the credentials discovered to connect to the group.

The test implementation experienced the same issues regarding undesirable NSD behavior mentioned in Wong et al.'s original paper [8]. In some instances, one or more of the test devices did not discover NSD broadcasts from other devices until they had been power cycled. The client then sees the already existing WiFi Direct group, but is unable to identify it as a group offering the chat service and to obtain the PSK required to connect. It then forms and advertises its own group resulting in two isolated groups in the same area competing for members.

B. Messaging

Using a message format with significant additional data results in additional data transfer. Most of the overhead comes from transmitting two full public keys (sender and recipient) with every message. II summarizes the size of a single message packet of various key sizes assuming 32 bit time stamps and fixed cryptographic parameters with compressed keys. The transmitted message contains 29 bytes of data, representing a typical instant message [16].

VII. Discussion

Restoring connectivity by using WiFi Direct in the event of an Internet outage does come with set up time, but may still be suitable for asynchronous applications such as IM and file transfers.

Issuing credentials to users in the form of digital certificates for use in asymmetric cryptography enables them to successfully authenticate one another with TLS during out-of-coverage operation. However, as it requires the user to be online at the time of sign up so that their digital certificate can be signed by the authentication component it might be a hindrance that prevent a new user from starting to use the application.

RSA Key Strength	Message Size	EC Curve	Message Size
2048 bit	1028 bytes	secp224r1	192 bytes
4096 bit	2052 bytes		
		secp384r1	292 bytes

TABLE II
MESSAGE PACKET SIZE IN BYTES GIVEN VARIOUS KEY SIZES SORTED BY STRENGTH. THE NAMED CURVES ARE SPECIFIED IN SEC 2 [17].

The overhead introduced by using strong cryptography is not insignificant, but this can be mitigated by choosing a cryptosystem with smaller key sizes.

If the TLS connection was not terminated at the GO, but GMs were allowed direct connections to each other, some of the application layer measures designed to protect the client from a dishonest server could have been avoided. However, this would have added to the implementation complexity, as the GO acting as a server makes out-of-coverage operation very similar to online operation. The fact that the server component code base can be used on both the dedicated server and on the GO makes the proposed system easy to adopt, but requires an application layer chat protocol.

The basic chat protocol that has been proposed provides confidentiality, integrity, authentication and is resilient against messages arriving out-of-order, but lacks more sophisticated properties that further protect users in edge cases such as private key compromise.

The chat protocol protects the contents of messages from a malicious GO, but the system does not, in its current state, attempt to detect a Denial of Service (DoS) type attack where the GO refuses to forward messages. Detecting this and electing a new GO automatically would make the system more resilient and useable for example in the case where a current Internet outage is a deliberate act aimed at disturbing communications. We leave the investigation of these edge cases as future work enhancements.

As WiFi Direct supports the same speeds as typical WiFi it is assumed that it provides more than sufficient throughput to carry a significant volume of these messages. In online mode, however, a server might have a much larger number of connected users. Keeping the message size low is therefore in the best interest of a developer to minimize the costs associated with bandwidth. Sending full public keys of both the sender and recipient with every packet adds significant overhead that could have been avoided by a different chat protocol design.

VIII. CONCLUSION

When smartphone applications are unable to connect to the Internet, many useful services become unavailable. In some cases, these services can be restored by communicating with nearby devices in a P2P fashion. However, services that exploit such technologies need to ensure security in the communication between nearby devices.

This work proposes an approach that ensures authenticated communication with confidentiality and integrity protection among peers. The approach enables devices to fetch security credentials in the form of digital certificates for use in asymmetric cryptography during sign up to a service. When unable to reach the centralized server, devices may use WiFi Direct to discover and connect to nearby devices, and mTLS to authenticate them while also setting up a secure data channel. A messaging protocol that may rely on the user's digital certificates must be used to protect message confidentiality and integrity during transfer over this channel.

This proposed design has been validated through implementation of a basic chat application for the Android OS. It makes use of a simple chat protocol that protect message confidentiality and integrity provided that the user's credentials are not compromised at any point in time. This has been validated by examining the transmitted data at various layers of the OSI model. The chat protocol lacks some advanced cryptographic properties such as forward secrecy, raising the question of whether or not current state of the art chat protocols can be adapted to a P2P scenario.

The main drawback to the proposed system is that the user is required to be online at the time of sign up, and that the user must trust the centralized server to be honest and not issue false credentials.

ACKNOWLEDGMENT

This research was funded by the joint EU FP7 Marie Curie Actions Cleansky Project, Contract No. 607584.

REFERENCES

[1] Ericsson, "Ericsson Mobility Report," november 2018.
[2] Wi-Fi Alliance, "Wi-Fi Simple Configuration Technical Specification v2.0.6," 2018.
[3] D. Camps-Mur, A. Garcia-Saavedra, and P. Serrano, "Device-to-device communications with wi-fi direct: overview and experimentation," *IEEE wireless communications*, vol. 20, no. 3, pp. 96–104, 2013.
[4] P. Shadbolt, "FireChat in Hong Kong: How an app tapped its way into the protests," *CNN*, 2014.
[5] "Briar User Manual," accessed September 12, 2019. [Online]. Available: https://briarproject.org/manual/
[6] P. Gardner-Stephen and S. Palaniswamy, "Serval mesh software-wifi multi model management," in *Proceedings of the 1st International Conference on Wireless Technologies for Humanitarian Relief*. ACM, 2011, pp. 71–77.
[7] A. A. Shahin and M. Younis, "A framework for P2P networking of smart devices using wi-fi direct," in *IEEE PIMRC 2014*.
[8] P. Wong, V. Varikota, D. Nguyen, and A. Abukmail, "Automatic android-based wireless mesh networks," *Informatica*, vol. 38, no. 4, 2014.
[9] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile," IETF, RFC 5280, May 2008.
[10] S. A. Vanstone, "Elliptic curve cryptosystem — the answer to strong, fast public-key cryptography for securing constrained environments," *Information security technical report.*, vol. 2, no. 2, pp. 78–87, 1997.
[11] V. G. Martínez, L. H. Encinas, and C. S. Ávila, "A survey of the elliptic curve integrated encryption scheme," *Journal of Computer Science and Engineering*, vol. 2, pp. 7–13, 01 2010.
[12] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ecdsa)," *International journal of information security*, vol. 1, no. 1, pp. 36–63, 2001.
[13] J. L. MacMichael, "Auditing wi-fi protected access (wpa) pre-shared key mode," *Linux J.*, vol. 2005, no. 137, pp. 2–, Sep. 2005.
[14] Y. Sheffer, R. Holz, and P. Saint-Andre, "Summarizing known attacks on transport layer security (TLS) and datagram TLS (DTLS)," IETF, RFC 7457, 2015.
[15] ——, "Recommendations for secure use of transport layer security (tls) and datagram transport layer security (dtls)," IETF, RFC 7525, 2015.
[16] R. Ling and N. S. Baron, "Text messaging and im: Linguistic comparison of american college data," *Journal of language and social psychology*, vol. 26, no. 3, pp. 291–298, 2007.
[17] Certicom Research, "SEC 2: Recommended elliptic curve domain parameters," in *Standards for Efficient Cryptography*, 2000.