Exposé - Analyzing Privacy Protection of Anonymous Communication Networks such as Vuvuzela and Stadium

Christoph Coijanovic

November 2019

Problem Statement

This thesis will analyze what level of privacy protection anonymous communication networks (ACN), such as *Vuvuzela* [1], *Stadium* [2], *Karaoke* [3] and *Yodel* [4] provide in a formal and standardized way with the framework proposed in [5]. It will also present a comparison between the ACNs based on the findings of the analysis.

Motivation

Today, the internet plays an integral role in nearly every person's life. Massive amounts of personal data are shared and stored, which makes privacy a vital issue. While today a majority of web traffic is encrypted and transmitted over HTTPS[1], *metadata*, such as the sender, receiver and time of the communication is often still leaked. This is where *anonymous communication networks (ACN)*, such as Tor[2], come in: They aim to not only protect the communication's content, but also the metadata.

Our goal is to provide a useful comparison between different ACNs, so that a privacy concerned person (e.g. a political activist or a journalist) can choose the best system for their individual needs. This task is complicated by the fact that the ACNs often define their own adversary models and privacy notions. [5] proposed formal privacy notions and a framework, which we will use to analyze various ACNs. We focus our analysis on the following ACNs:

[1]See Google Transparency Report: https://transparencyreport.google.com/https/overview
[2]https://torproject.org

- **Vuvuzela** [1], since it claims to be the first private messaging system that can hide metadata while scaling to more than a few thousand users[3]
- **Stadium** [2], since it promises to be able to handle four times the number of users of Vuvuzela[4]
- **Karaoke** [3], since it promises even better scalability and latency than Vuvuzela and Stadium[5]
- **Yodel** [4], since it is an ACN capable of real-time communication, which requires a significantly lower latency than the other systems

One important question that the thesis will try to answer is if the improvements to scalability and latency of Stadium, Karaoke and Yodel come at the cost of privacy in comparison to Vuvuzela.

Background

We will first discuss different kinds of adversaries, then examine how ACNs in general work using *Vuvuzela* as an example and finally take a brief look at the framework from [5].

On Adversaries

The following section on common adversary models is based on [6]. When making statements about the level of protection that an ACN provides, one has to specify what kind of adversary is considered. Imagine a network where messages are send from sender to receiver through a chain of multiple servers. To ensure message confidentiality, the sender encrypts the message content with the first server's public key and sends the ciphertext to the server. The server decrypts it, re-encrypts it with the public key of the next server, forwards it and so on. In such a network, the content of messages would be hidden to an adversary, that can only passively observe the traffic on the network links, since only encrypted messages are sent. But as soon as the adversary is able to *corrupt* one of the servers in the chain (e.g., by acquiring its secret key), he can find out the message plaintext.

[6] proposes four main abilities an attacker can have:

- **Passive Observation** (G): The adversary can observe the activity of nodes and on links. He can do so either globally (on all nodes/links) or restricted to a subset of nodes/links.
- **Active Corruption** (C): The adversary can corrupt a defined number of links and nodes.

[3]see introduction of [1]
[4]see abstract of [2]
[5]see abstract of [3]

- **Timing** (T): The adversary has access to a timer which he can use for passive traffic analysis.
- **Traffic Modification** (M): The adversary can actively modify the network traffic, i.e. insert, delete or modify packets.

A concrete adversary can be created by combining the abilities above. For example, $\mathcal{A}_{GCM}$ is an attacker that can observe the network, actively control (a part of) the network and modify the traffic, but cannot delay packets since no timer is available.

Vuvuzela

We chose *Vuvuzela* as an example, because it was the first ACN that could scale to more than a few thousand users [1, Sec. 1]. It also shares many of the basic mechanisms with the ACNs that came after it while being architecturally relatively simple. Vuvuzela's network (see Figure 1) consists of three parts: Users, the mixnet and dead drops. The mixnet are a number of servers connected in a line. Users send their messages to the first server, they get passed from server to server until the last puts the messages in their corresponding dead drop. Dead drops are basically postboxes that are used to exchange messages between pairs of users.

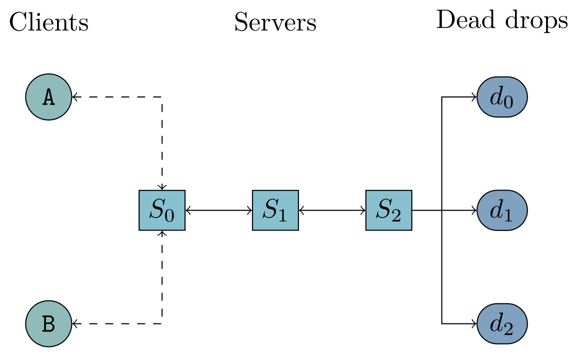

Figure 1: Vuvuzela network diagram

Communication in Vuvuzela occurs in rounds. The first server coordinates the rounds by broadcasting the beginning of a new round to all clients and starting a countdown during which clients can submit their messages. If Alice wants to send a message to Bob in round n, she has to compute shared round keys with Bob (k_{AB}) and every server (k_{AS_i}) in the mixnet (based on her and Bob's/the servers public-keys) and the id of their shared dead drop (based on their shared round key and the current round number). Then she can generate a request e containing the dead drop id and her message encrypted with k_{AB}. Then she *onion wraps* her request as follows: $Enc_{k_{AS_0}}(Enc_{k_{AS_1}}(\ldots Enc_{k_{AS_m}}(e)))$ and sends it to the first server (S_0). If Alice does not have a "real" message to send, her client automatically sends a cover message to hide her activity[6]. Every server in the mixnet first waits for all messages in this round to arrive, then unwraps the top layers of the requests, generates random cover traffic[7] and shuffles all requests. If the current server is not the last in line, it sends the shuffled requests to the next server in line. If the current server is the last server it matches the messages to the corresponding dead drop. If there was a pair of exchanges on the same dead drop, the server will exchange the content of those two requests and send them back through the mixnet to the clients. While requests return to their sender, the onion layers are build up step by step in each server they pass.

The Framework

[5] introduces a game model that is very similar to the IND-CPA game used in formal security proofs of cryptographic systems. An adversary $\mathcal{A}$ has to submit a *challenge* consisting of two *scenarios* $\underline{r}_0$ and $\underline{r}_1$ (a scenario consists of multiple *communications*, each defined by sender, receiver, message and auxiliary information) to the challenger $\mathcal{C}$. $\mathcal{C}$ then chooses one of those scenarios at random by choosing a single bit b uniformly at random and submitting $\underline{r}_b$ to the ACN Π. $\mathcal{A}$ receives all output from Π and has to decide which of his scenarios was submitted to Π by $\mathcal{C}$. He does so by returning a single bit b' which represents his guess on what value $\mathcal{C}$ chose for b was. $\mathcal{A}$ wins if $b' = b$ and loses otherwise. An overview of this game can be found in Figure 2.

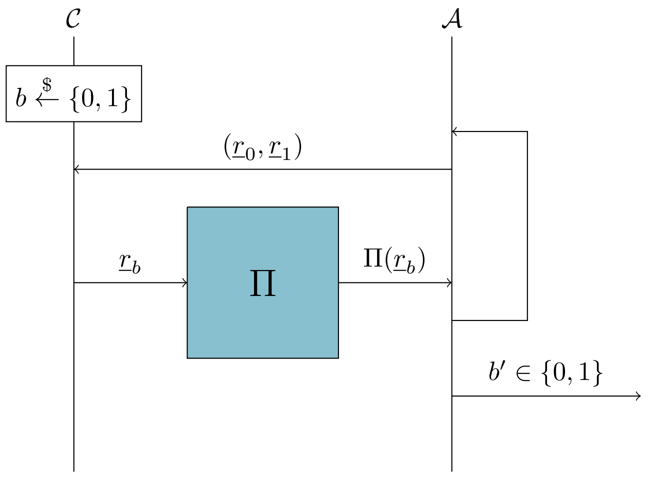

Figure 2: The formalized security game. $\mathcal{A}$ wins if $b' = b$.

[6]If Alice is in an active conversation, the (empty) cover message is send to this recipient, else a random public key is generated and used as the recipient.

[7]It seems like cover traffic is send along the line to last server and not removed by the next server (so the total number of requests increases at each intermediate server). On the way back, every server seems to remove the cover traffic that it added.

This game then can be used to prove that a given ACN fulfills a given *privacy notion*. As an example, a notion an ACN could have is *sender-message unlinkability*. If an ACN achieves this notion, attackers cannot distinguish which message was send by which sender[8]. To model this notion with the game, we restrict the adversary to submit scenarios that only differ in the senders. Everything else (e.g., message content, the number of send messages, the receiver and the auxiliary information) has to be the same (see Fig. 3).

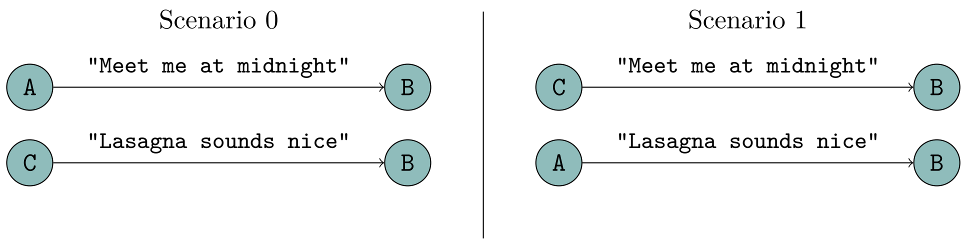

Figure 3: An example for an allowed challenge for sender-message unlinkability

If it can be shown that there is no adversary that can win this game for a given ACN, the ACN achieves sender-message unlinkability. If an adversary can be found which can win the game despite the restrictions, the ACN is proven not to fulfill sender-message unlinkability.

Our Contribution

Our Approach

We want to compare the level of privacy protection that different ACNs offer by analyzing them using [5]'s framework as a base. We plan to analyze the ACNs ordered by their publication date (i.e. Vuvuzela, Stadium, Karaoke, Yodel), since the later systems build on the techniques and findings of the earlier ones. We will base our approach for analysis on the "How to use" suggestion in [5, Sec. 11]:

1. Select an adversary to analyze against. We will start with the attacker suggested in the source paper[9] (for Vuvuzela, that would be an attacker that can control all but one server, an arbitrary number of clients and can monitor, block, delay, or inject traffic at every network link [1, Sec. 2.3]). If we find that analyzing the ACN with another attacker in mind will aid our comparison, we will do so as an additional step.

[8]He might still be able to read the content of the messages or find out who the message was send to, but the identity of the sender remains hidden.

[9]if sufficiently defined

2. Create a simplified version of the ACN protocol (ideal functionality) and proof that is indistinguishable from the real protocol
3. Extract properties from the ideal functionality
4. Find out which privacy notions the protocol achieves. We start with the strongest possible notions and try to disproof that the ACN achieves this notion. A notion is not achieved, if the adversary can learn a property by playing the game described in Figure 2 that he is not allowed to learn for the given notion. If we are not able to disproof that the notion is achieved, we will try to proof it (see [7])
5. Compare the ACN to the previously analyzed protocols. If there are differences in the achieved notions, find out what protocol mechanisms caused them.

Solving The Problem

The main problem when comparing ACNs is the varying definitions and adversary models. By using a single framework on all of them, we remove this variance and bring them all on common ground. Since the notions introduced in [5] have a hierarchy associated with them, once we have determined which ACNs fulfill which notions, we can make concrete statements about their privacy levels in relation to each other.

Evaluation

To evaluate our results, one has to know the assumptions we make. In our analysis we will make the following assumptions[10]:

1. There is at least one server in the network that is not corrupted by the adversary
2. All non-corrupted clients and servers follow the protocol faithfully
3. There are no bugs in the implementation that leak information that would otherwise not have leaked
4. The adversary can not break the utilized cryptography
5. There is no user error (e.g. users leaking their secret keys by accident)

Timeplan

- Time frame: 01.01.2019 to 30.06.2020 = **26 Weeks**
- **Before week 0**: Preparation, understand the framework, do first analysis exercises
- **Weeks 1 to 8**: Vuvuzela and exams

[10]Further assumptions might need to be made during the analysis

    - 1-2: Understand Vuvuzela and create simplified protocol
    - 3-4: Extract properties
    - 5-8: Find out what notions are achieved (and prepare for 2 oral exams[11])
- **Weeks 9 to 14** Stadium
    - 9-10: Understand Stadium and create simplified protocol (and prepare for one exam on 10.03.20)
    - 11: Extract properties
    - 12-13: Find out what notions are achieved
    - 14: Comparison to Vuvuzela
- **Weeks 15 to 18**: Karaoke
- **Weeks 19 to 22**: Yodel
- **Week 23**:
    - compare all analyzed ACNs
    - prepare conclusion
- **Weeks 24 to 26**:
    - compile all findings
    - finalize thesis

Table 1: Overview of the timeplan

Topic	Time frame
Vuvuzela	01.01. - 23.02.
Stadium	24.02. - 05.04.
Karaoke	06.04. - 03.05.
Yodel	04.05. - 31.05.
Comparison	01.06. - 07.06.
Finalization	08.06. - 30.06.

References

[1] J. van den Hooff, D. Lazar, M. Zaharia, and N. Zeldovich, “Vuvuzela: Scalable private messaging resistant to traffic analysis,” in *Proceedings of the 25th symposium on operating systems principles*, 2015, pp. 137–152 [Online]. Available: http://doi.acm.org/10.1145/2815400.2815417

[2] N. Tyagi, Y. Gilad, D. Leung, M. Zaharia, and N. Zeldovich, “Stadium: A distributed metadata-private messaging system,” in *Proceedings of the 26th symposium on operating systems principles*, 2017, pp. 423–440 [Online]. Available: http://doi.acm.org/10.1145/3132747.3132783

[3] D. Lazar, Y. Gilad, and N. Zeldovich, “Karaoke: Distributed private messaging

[11]exact dates TBD.

immune to passive traffic analysis," in *13th {usenix} symposium on operating systems design and implementation ({osdi} 18)*, 2018, pp. 711–725.

[4] D. Lazar, Y. Gilad, and N. Zeldovich, "Yodel: Strong metadata security for real-time voice calls."

[5] C. Kuhn, M. Beck, S. Schiffner, E. Jorswieck, and T. Strufe, "On privacy notions in anonymous communication," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 2, pp. 105–125, Apr. 2019 [Online]. Available: http://dx.doi.org/10.2478/popets-2019-0022

[6] C. Kuhn and M. Beck, "Common adversaries," *Technical report*, 2019.

[7] N. Gelernter and A. Herzberg, "On the limits of provable anonymity." *IACR Cryptology ePrint Archive*, vol. 2013, p. 531, 2013.